Analysis of the Legendre Random Number Generator

Martin Rupp
Peter Smirnov

SCIENTIFIC AND COMPUTER DEVELOPMENT SCD LTD

Contents

1 Finite fields of characteristic 2

1.1 Squares in finite fields of characteristic =2

Firstly let's suppose a finite field of characteristic 2 , $F_{2^s} = GF(2^s)$. Any element in that field is a square. We check: for example $GF(8)$. The elements of $GF(8)$ are :

$g_0 : P(X) = 0$
$g_1 : P(X) = 1$
$g_2 : P(X) = X$
$g_3 : P(X) = X + 1$
$g_4 : P(X) = X^2$
$g_5 : P(X) = X^2 + 1$
$g_6 : P(X) = X^2 + X$
$g_7 : P(X) = X^2 + X + 1$

They are considered as elements of $K[X]/(X^3 + X + 1)$ where K is the field $Z/2Z$

The application $g \to g^2$ from $GF(8)$ into itself is a bijection:

$g_0^2 = g_0$
$g_1^2 = g_1$
$g_2^2 = g_4$
$g_3^2 = g_5$
$g_4^2 = g_6$
$g_5^2 = g_7$
$g_6^2 = g_2$
$g_7^2 = g_3$

To prove that $\varphi : g \to g^2$ is always a bijection for any group $GF(2^s)$, we remark that φ is an endomorphism since $\varphi(g + h) = g^2 + h^2 = \varphi(g) + \varphi(h)$ in the specific case of the fields of characteristic two. In fact,φ is the Frobenius Endomorphism.

$Ker(\varphi)$=$\{g, g^2 = 1\}$, since $Ker(\varphi)|F_{2^s}^*$, then $|Ker(\varphi)|$ must be odd.
If $Ker(\varphi)$ has an element g_c which is of order 2 ($g_c^2 = 1$) then this means that $2|Ker(\varphi)$ which is not possible, hence $Ker(\varphi)$ has only elements of order 1, e.g φ is an automorphism.

Elements of $GF(2^s)$ has only a square root since $-g = +g$ in characteristic 2.
Finally $GF(2^s)$ is always cyclic so there exist at least one generator ξ such that all elements can be written as a power $g = \xi^i$, hence to find a square root of g, we need to find k such that $2k = i(mod 2^s - 1)$ and $\sqrt{g} = \xi^k$.

In the example of $GF(8)$, we form the complete multiplication table of its elements:

	0	1	g_2	g_3	g_4	g_5	g_6	g_7
0	0	0	0	0	0	0	0	0
1	-	1	g_2	g_3	g_4	g_5	g_6	g_7
g_2	-	-	g_4	g_6	g_3	1	g_7	g_5
g_3	-	-	-	g_5	g_7	g_4	1	g_2
g_4	-	-	-	-	g_6	g_2	g_5	1
g_5	-	-	-	-	-	g_7	g_3	g_6
g_6	-	-	-	-	-	-	g_2	g_4
g_7	-	-	-	-	-	-	-	g_3

We see that , for instance, g_2 is a generator of the group since $g_2^7 = 1$ and $g_2^x \neq 1, x < 7$.

$g_2^1 = g_2$
$g_2^2 = g_4$
$g_2^3 = g_3$
$g_2^4 = g_6$
$g_2^5 = g_7$
$g_2^6 = g_5$

If we want to compute $\sqrt{g_3}$ we need to find k such as $2k \equiv 3(Mod7)$ since $g_3 = g_2^3$ and we will have $\sqrt{g_3} = g_2^k$. we find that k=5 and $\sqrt{g_3} = g_2^5 = g_7$.

Let's check that for the multiplicative group F_q^* $(q = p^s)$ of any finite commutative field F_q, we can always find a generator ξ:

If $|F_q^*| = n$ and $d|n$ then we consider the set of elements of order d: $H(d) = \{g \in F^*, g^d = 1, g^e \neq 1, e < d\}$. Let $f(d) = |H(d)|$, then $f(d) \leq \psi(d)$ where ψ is Euler's function.

Because F_q is a field, the set $H(d)$ cannot have more than d elements. Indeed the equation $X^d - 1 = 0$ has at most d distinct roots in F_p.

There are two cases to consider:

1. $f(d) = 0$. In such case we have $f(d) \leq \psi(d)$;
2. $f(d) > 0$. In such case we can find an element $g \in H(d)$. The cyclic subgroup G_d generated by g is group-isomorphic to Z/dZ and it has exactly d elements. But since every element x of G_d is such that $x^d = 1$ (e.g the elements $\{1, g, g^2, ..., g^{d-1}\}$) then G_d regroups the totality of the elements of F_p^* that verify $x^d = 1$ and then $H(d)$ is contained into G_d: $H(d) \subseteq G_d$. This means that $H(d)$ is in fact the set of elements of order d of G_d, so in such a case, we can use the well-known formula that gives the amount $N(e)$ of elements of order e in a cyclic group of order d, namely: $N(e) = \psi_d(e)$.

As a special case e=d, we have:

$$f(d) = \psi(d).$$

Since we have a partition $F_q^* = \coprod_{d|n} H(d)$, we get:
$n = |F_q^*| = \sum_{d|n} f(d)$
but since we also have $n = \sum_{d|n} \psi(d)$ we must have $f(d) = \psi(d)$ for all the divisors d of n. This also implies that $f(n) = \psi(n) > 0$ and therefore that there exists generator of F_q^*.□

2 Finite fields of characteristic >2

2.1 Squares in finite fields of characteristic >2

If $p > 2$ and $q = p^s$ then it's no longer true that all elements of the finite field F_q has a square root in F_q.
In fact in such case, the set of squares of F_q^* form a subgroup of index 2 in F_q. This means that there are exactly $(|F_q| - 1)/2$ non zero elements that has a square root in F_q (recall that since p>2, $(-1)^q = -1$).

If we consider elements y of an algebraic closure $Cl(F_q)$ of F_q such that $y^2 = x$, for $x \in F_q^*$, then since we have $x^{q-1} = 1$ (the order of x must be a divisor of q-1) , this implies that $(x^{(q-1)/2})^2 = 1$and therefore $x^{(q-1)/2} = \pm 1$.That is to say $y^{q-1} = \pm 1$.

So if x is a square, then $y \in F_q$ and $y^{q-1} = 1$, e.g $x^{(q-1)/2} = +1$.
And if $x^{(q-1)/2} = -1$ then $y^{q-1} = -1$ then $y \notin F_q^*$ and x cannot be a square.

This shows that the set of squares is the kernel of $f : x \to x^{(q-1)/2}$ and is therefore a group F^{*2}_q.

$Ker(f)$ is also the set of elements of F_q^* of order $(q-1)/2$ or less and there are exactly $\sum_{d|(q-1)/2} \psi(d) = (q-1)/2$ such elements (as we saw previously in the multiplicative group of a finite field) . Therefore the index of F^{*2}_q in $F^*{}_q$ is $= \frac{(q-1)}{(q-1)/2} = 2$.

In order to know if an element x of the finite field is a square or not, we have to study the sign of the quantity $f(x) = x^{(q-1)/2}$. In the case of q=p (recall here we are in the case p>2) , e.g when the finite fields is $Z_p = Z/pZ$, that function is formally named the *Legendre symbol* and is noted $(\frac{x}{p})$.

We have $(\frac{x}{p})(\frac{y}{p}) = (\frac{xy}{p})$ for any two elements $(x, y) \in Z_p$ with the additional convention that $(\frac{0}{p}) = 0$
In such a case x is a square in Z_p if and only if its legendre symbol is equal to +1. In such a case, the terminology is that x is *a quadratic residue modulo p.*

2.2 Computation of the Legendre Symbol

2.2.1 Cases x=-1,+1,2

If we define two following additional symbols ε, ω:
$\varepsilon(n) \equiv (n-1)/2 \pmod 2$ and $\omega(n) \equiv (n^2-1)/8 \pmod 2$, for n odd.
Then we can compute the Legendre symbol for $x = +1, -1, p$ by:

$(\frac{1}{p}) = 1^{(p-1)/2} = 1$
$(\frac{-1}{p}) = (-1)^{(p-1)/2} = (-1)^{\varepsilon(p)}$

The computation of $(\frac{2}{p})$ is a bit less trivial:

We have $(\frac{2}{p}) = 2^{(p-1)/2}$

Let's check that in fact if we name $\nu(2,p)$ the amount of elements from the set $\{2,4,6,...,p-1\}$ which are greater than $(p-1)/2$, then we have $(\frac{2}{p}) = (-1)^{v(2,p)}$.

To see this, let's visualise the sequence $E_p = \{2,4,6,...,2k,...,(p-1)(p)\}$.
And the sequence $E_P \pmod p$ of these elements, modulo p.

$E_P \pmod p$ is in fact $\{\epsilon_i \sigma(i)\}_{1 \leq i \leq (p-1)/2}$ where σ is a permutation of the set $\{1,..,(p-1)/2\}$ and ε a function from that set to $\{-1,+1\}$

To see that it is a permutation, we suppose that there exists $i \neq j$ such that $\epsilon_i \sigma_i = \epsilon_j \sigma_j$.
We must have $\epsilon_i \epsilon_j = -1$ so for instance $\sigma_i = -\sigma_j$.

In such a case, we would have $2(i+j) = 0 \pmod p$ but since $i + j \leq p-1$, this is impossible.

If we take the product of both E_p and $E_p \pmod p$, we get:

$$\prod_{i=1}^{i=(p-1)/2} 2i = \prod_{i=1}^{i=(p-1)/2} \epsilon_i \sigma_i \pmod p$$

This leads to:

$$(\tfrac{2}{p})((p-1)/2)! = (\prod_{i=1}^{i=(p-1)/2} \epsilon_i) \ \ ((p-1)/2)! \pmod p$$

$$((\tfrac{2}{p}) - (\prod_{i=1}^{i=(p-1)/2} \epsilon_i) \ \ ) \times ((p-1)/2)! \ = 0 \pmod p$$

And finally, since p do not divide $((p-1)/2)!$, to:

$$(\tfrac{2}{p}) = (-1)^{v(2,p)} \pmod p$$

That is to say:

$(\frac{2}{p}) = (-1)^{v(2,p)}$

Where $v(2,p)$ is the number of negative values of ϵ_i for $i = 1,...,(p-1)/2$

That number is also the amount of even numbers present in the set $](p-1)/2, p-1]$ $\square$

It is therefore straightforward to check that the amount of such even numbers is given by the function $\omega(p)$ and hence $(\frac{2}{p}) = (-1)^{\omega(p)}$.

That demonstration is in fact based on a special case of Gauss's Lemma (see after)

There is an alternate and simpler way of computing $(\frac{2}{p})$ by using a primitive 8^{th}-root of unity ξ. Indeed $y = \xi + \xi^{-1}$ is such that $y^2 = \xi^2 + \xi^{-2} + 2 = 2$.

$(\xi^8 = 1 \Rightarrow \xi^4 = \pm 1 \Rightarrow \xi^4 = -1 \Rightarrow \xi^2 + \xi^{-2} = 0)$

In such a case $(\frac{2}{p}) = y^{(p-1)}$

We also have $y^p = \xi^p + \xi^{-p}$ (Freshman's dream)

If $p = s \pmod 8$, we have $(\frac{2}{p}) = \xi^s + \xi^{-s}$

For $s = \pm 1$, this implies $y^p = y$ and hence $(\frac{2}{p}) = +1$.

For $s = \pm 5$, this implies $y^p = \xi^4 \xi^{s'} + \xi^4 \xi^{-s'}$ where $s' = \pm 1$

$y^p = -y$ and hence $(\frac{2}{p}) = -1$.□

2.2.2 General cases

The law of quadratic reciprocity links two 'reciprocal' legendre symbols as such:

Let (p, q) be two different odd primes, then:

$(\frac{q}{p}) = (\frac{p}{q})(-1)^{\varepsilon(p)\varepsilon(q)}$

There are several demonstrations. The simplest way is consider the Gauss sum :

$S = \sum_{x \in F_q} (\frac{x}{q}) \xi^x$ where ξ is a q^{th} primitive root of unity. Since we can identify F_q with Z/qZ, the element ξ^x is defined for $x \in F_q$.

Then it is only needed to prove that :

1. $S^2 = (-1)^{\varepsilon(q)} q$

2. $S^{p-1} = (\frac{p}{q})$

Indeed in such a case, we have almost immediately that

$$(\frac{p}{q}) = ((-1)^{\varepsilon(q)}q)^{(p-1)/2} = (\frac{q}{p})(-1)^{\varepsilon(p)\varepsilon(q)}.\square$$

1. $S^2 = [\sum_{x\in F_q}(\frac{x}{q})\xi^x] \times [\sum_{x\in F_q}(\frac{x}{q})\xi^x] = \sum_{(x,y)\in F_q}(\frac{x}{q})(\frac{y}{q})\xi^{x+y} = \sum_{(x,y)\in F_q}(\frac{xy}{q})\xi^{x+y}$

$S^2 = \sum_u C_u\xi^u$ with $C_u = \sum_t(\frac{t(u-t)}{q})$ (the sums are over all F_q)
$C_u = \sum_{t\neq 0}(\frac{-t^2(1-u/t)}{q}) = \sum_{t\neq 0}(\frac{-t^2}{q})(\frac{1-u/t}{q}) = (\frac{-1}{q})\sum_{t\neq 0}(\frac{(1-u/t)}{q}) = (\frac{-1}{q})C'_u$
$C'_0 = \sum_{t\neq 0}(\frac{1}{q}) = \sum_{t\neq 0} 1 = q-1$
If $u \neq 0$:
$C'_u = \sum_{t\neq 0}(\frac{(1-u/t)}{q}) = \sum_{v\neq 1}(\frac{v}{q}) = -1$ since $\sum_{v\neq 1}(\frac{v}{q}) = 0$
So we have , at the end:

$S^2 = (\frac{-1}{q})\sum_u C'_u\xi^u = (\frac{-1}{q})(q-1-\sum_{u\neq 0}\xi^u)$
And, since $\sum_{u\neq 0}\xi^u + 1 = 0$

$S^2 = (\frac{-1}{q}) \times q$

$\square$

2)$S^p = [\sum_{x\in F_q}(\frac{x}{q})\xi^x]^p = \sum_{x\in F_q}(\frac{x}{q})^p\xi^{px}$ (Freshman's dream)

$S^p = \sum_{z\in F_q}(\frac{z/p}{q})\xi^z = (\frac{p}{q})\sum_{z\in F_q}(\frac{z}{q})\xi^z = (\frac{p}{q}) \times S$

$\square$

Another demonstration consisting in using Gauss lemma can be found in [1] .

We recall Gauss lemma:
If $F_p = H \cup (-H)$ then $(\frac{a}{p}) = \prod_{s\in H} e_s(a)$, where $as = e_s(a)s_a, (s_a, s) \in H$
E.g as has coordinates (X, Y)in $\{\pm 1\} \oplus H$ and e_s is the projection operator of as on $\{\pm 1\}$.

3 Algorithmic Computations of the Legendre symbol

With the Law of quadratic reciprocity, the computation of the Legendre symbol is made possible recursively. For this we use the following two additional properties of the Legendre symbol:

1. $(\frac{x}{p})(\frac{y}{p}) = (\frac{xy}{p})$
2. $(\frac{x}{p}) = (\frac{z}{p})$ if $x \equiv z(mod p)$

For example we want to compute $(\frac{17}{43})$. we have :

$(\frac{17}{43}) = (-1)^{\varepsilon(17)\varepsilon(43)} \times (\frac{43}{17}) = (-1)^{0\times 1}(\frac{43}{17}) = (\frac{43}{17})$
$(\frac{43}{17}) = (\frac{9}{17}) = (\frac{3}{17})^2 = +1$

$(\frac{17}{43}) = 19,342,813,113,834,066,795,298,816(mod43) = +1$
And this mean that 17 is a square in $Z/43Z$.

The following procedure will allow us to compute any Legendre symbol:

1. Start procedure compute (x/p)
2. if(x>p) reduce x modulo p so that x<p.

1. Test x for primality using primality tests
2. If the test is concluent use the quadratic reciprocity law
   to compute (p/x) and move to (1)
3. If the test is not concluent there is a divider d of x: x=dy
4. Compute (d/p)(y/p) move to (1)

Here is an implementation in C#:

```
using System;
using System.Numerics;

namespace legendre
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0;
            int p = 0;
#if false
            while (true)
            {
                Console.WriteLine("x?");

                Int32.TryParse(Console.In.ReadLine(), out x);

                Console.WriteLine("p?");

                Int32.TryParse(Console.In.ReadLine(), out p);
```

```
            Console.WriteLine("(" + x + "/" + p + ")=" + Legendre(x, p));
        }
#endif
 int[] primes = new int[] { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191,
193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311,
313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439,
443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577,
587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709,
719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857,
859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997 };

 Random r = new Random();
        int error = 0;
        BigInteger power;
        for (int i = 0; i < 1000; i++)
        {
            x = r.Next(1, 100);
            foreach (int p1 in primes)
            {
                while(x%p1==0)
                    x = r.Next(1, 100);
                int L1 = Legendre(x, p1);
                power = BigInteger.Pow(x, (p1 - 1) / 2);
                BigInteger L2 = BigInteger.Remainder(power, p1);
                if (L2 > 1)
                    L2 = L2-p1;
                Console.WriteLine("**************************");
                Console.WriteLine("(" + x + "/" + p1 + ")=" + L1);
                Console.WriteLine("(" + x + "/" + p1 + ")=" + L2);
                if (L1 != L2)
                    error++;
            }
        }
        Console.WriteLine("errors=" + error);

        Console.ReadLine();
    }
    static int e(int x)
    {
        return ((x - 1) / 2 ) % 2;
    }
    static int O(int x)
    {
        return ((x*x - 1) / 8) % 2;
    }

    static int Legendre(int x, int p)
    {

        if (x > p)
            x = x % p;
```

```
            if (x == 0)
                return 0;
            if (x == 1)
                return 1;
            if (x == -1)
                return (int)(Math.Pow(-1, e(p)));
            if (x == 2)
                return (int)(Math.Pow(-1, O(p)));
//for big primes need a probabilistic primality test
            for (int d=2;d<(x/2)+1;d++)
            {
                if(x%d==0)
                {
                    return Legendre(d, p) * Legendre(x/d, p);
                }
            }
            //prime
            return ((int)(Math.Pow(-1, e(x) * e(p)) )* Legendre(p, x);
        }

    }
}
```

When we run the program we can check that the algorithm works well:

168000 legendre symbols computed
errors=0

4 Jacobi Symbol

The Jacobi symbol, noted $(\frac{a}{n})$ is a natural extension of the Legendre symbol to all the natural numbers $n \in N$ by multiplicativity :

If $n = \prod_i p_i^{v_i}$is the decomposition of n in prime factors then:

$(\frac{a}{n}) = \prod_i (\frac{a}{p_i})^{v_i}$

5 Application to Random Number Generation

5.1 Legendre RNG

There are applications of the Legendre symbol to Random Number Generation.
In fact since there are exactly as much quadratic residues as non-quadratic residues in $(Z/pZ)^*$, and since they have naturally binary values, the legendre (and Jacobi) symbols are good candidates for a random generator.

If the sequence$L_a = (\frac{a}{p})$, $1 \leq a < p$, was truly random, then there is exactly a probability $1/2$ that it is $+1$ and a probability $1/2$ that it is -1. Given a parameter l, the probability for a random sequence of numbers from ± 1 S_a , of length l, to have a given value $E = \{\varepsilon_1, .., \varepsilon_l\}$, $\varepsilon_i = \pm 1, 1 \leq i \leq l,$ is:
$P(S_a = E) = \prod_{i=1}^{i=l}(1/2) = (1/2)^l$
And so the amount N of matching values $S_a = E$ for $1 \leq a < p$ is given by :
$N = (p-1)/2^l$
Since we have $(p-1)$ tries.

We can test how close to that value is an estimator $\hat{P}$ taken as

$$\hat{P} = \frac{1}{n}\sum_{i=1}^{i=n}(\text{amount of samples that are equal to } E_i \text{ / total amount of samples})$$

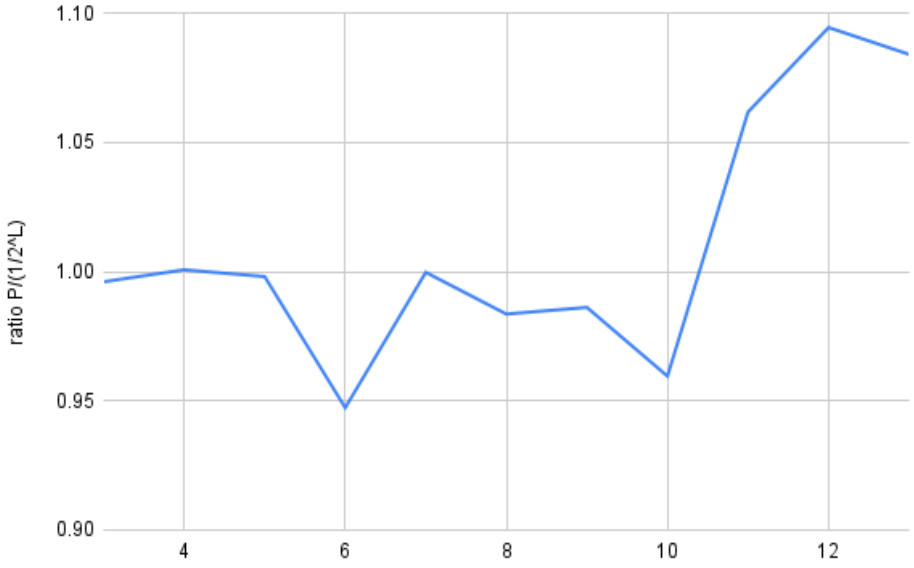
Where the fixed values E_i are randomly generated.

For a set of sequences $S_a = \{(\frac{a}{p}), ..., (\frac{a+l}{p})\}$ Since $\hat{P} \sim P(S_a = E)$.

Here are the values obtained for $p = 3089$ and 2^{15} samples:

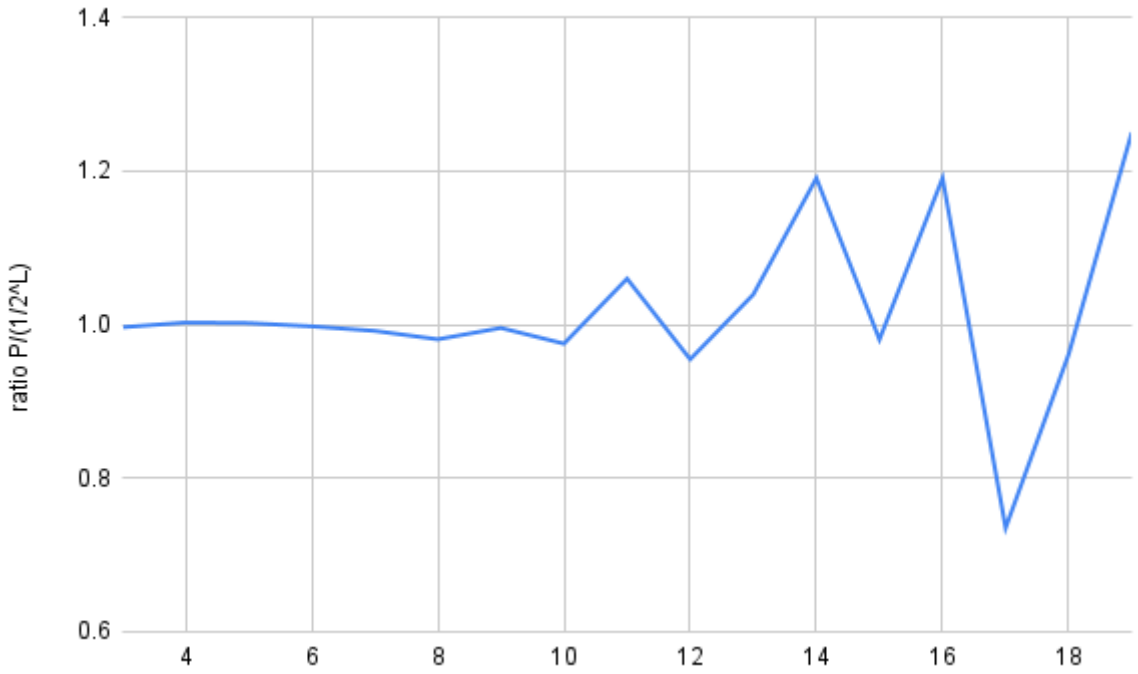
l	$\hat{P}$	$(1/2)^l$	$\frac{\hat{P}}{(1/2)^l}$
3	0.1254968262	0.125	0.9960411256
4	0.06245788574	0.0625	1.000674282
5	0.03131103516	0.03125	0.9980506823
6	0.01649414063	0.015625	0.9473060983
7	0.007814941406	0.0078125	0.9996875976
8	0.003971557617	0.00390625	0.9835561703
9	0.00198059082	0.001953125	0.9861325116
10	0.00101776123	0.0009765625	0.9595202399
11	0.0004598999023	0.00048828125	1.061712011

As a first basic observation, the ratio $\frac{\hat{P}}{(1/2)^l}$seems to converge towards 1.



It is needed to obtain a large ratio between the total space of the values which is $p-1$ and the amount of possible samples matching E.
Here are the values obtained for $p = 988303427$ and 2^{15} samples:

l	$\hat{P}$	$(1/2)^l$	$\frac{\hat{P}}{(1/2)^l}$
3	0.1254486084	0.125	0.9964239667
4	0.06234771729	0.0625	1.002442475
5	0.03119476318	0.03125	1.001770708
6	0.01566253662	0.015625	0.9976034137
7	0.007878417969	0.0078125	0.9916330958
8	0.003983154297	0.00390625	0.9806926142
9	0.001961975098	0.001953125	0.9954891896
10	0.001001586914	0.0009765625	0.9750152346
11	0.0004608154297	0.00048828125	1.059602649
12	0.0002557373047	0.000244140625	0.9546539379
13	0.0001174926758	0.0001220703125	1.038961039
14	0.00005126953125	0.00006103515625	1.19047619
15	0.00003112792969	0.00003051757813	0.9803921569
16	0.00001281738281	0.00001525878906	1.19047619
17	0.00001037597656	0.000007629394531	0.7352941176
18	0.000003967285156	0.000003814697266	0.9615384615
19	0.000001525878906	0.000001907348633	1.25



5.2 Analysis of the Legendre RNG randomness

5.2.1 Entropy

Entropy in the context of randomness measures the frequency of occurrence of characters, e.g "Shannon's Entropy".

If we use an alphabet with N symbols - say $U_1, ..., U_N$then the Shannon entropy $H(X)$ of a "word"X is:
$H(X) = -\sum_{i=0}^{p-1} p_i Log_2(p_i)$

Where p_i is the probability of appearance of the symbol U_i.

Here we will compute p_i by its frequency, e.g $p_i = \frac{S_i}{S}$ and $S_i$is the amount of occurences of the symbolU_i while $S$is the total amount of symbols in the word.
We also will consider that $0 * Log_2(0) = 0$.

The Shannon Entropy is a positive number which may be used to measure the randomness of a word. A maximal value for the entropy means that the word has "best" randomness.

$H(X) = -\frac{1}{S}\sum_{i=0}^{p-1} S_i Log_2(S_i/S)$
The function $f(x) : x \to -xLog_2(x)$is concave , therefore we have the following inequality:

$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \leq f(\frac{1}{N}\sum_{i=0}^{N-1} x_i)$

Or:

$$-\frac{1}{p}\sum_{i=0}^{p-1}(S_i/S)Log_2(S_i/S) \leq -(\frac{1}{p}\sum_{i=0}^{p-1} S_i/S) * Log_2(\frac{1}{p}\sum_{i=0}^{p-1} S_i/S)$$

Which leads to:

$H(x) \leq Log_2(p)$

This means that $Log_2(p)$is the maximal value for the entropy of a word with p symbols. How close to this value if the entropy computed from a generator is a measure of the quality of the randomness of this generator.

In the case of a byte generator, the closest the computed entropy from the RNG is from $Log_2(256) = 8$, the better the RNG is.

5.2.2 Basic Statistical Tests

Kolmogorov-Smirnov The Kolmogorov-Smirnov is a Goodness-of-Fit test. It measures how "close" is an experimental/observed distribution against an expected one.

Here of course, we wish to examine how close the distribution of the numbers produced by the generator are from the uniform distribution.

If we observe N samples: $X_1, ..., X_N$ then we compute the Kolmogorov-Smirnov statistics D_N to compare how these samples 'fits' with a distribution which has a cdf $F(x)$.

For this we compute the cdf for the observed distribution:

$F_N^*(x) = \sum_{X_i \leq x} 1$
$D_n = sup_x |F_N^*(x) - F(x)|$

In the case of a uniform law $Unif$(a,b) then, $F(x) = \frac{[x]-a+1}{n}$ with $n = b - a + 1$.

To accept the hypothesis null H_0 (goodness of fit) at a level of α we need to check the value K_α defined by $Prob(K \leq K_\alpha) = 1 - \alpha$ where K is the kolmogorov distribution ($K = sup_x |B(x)|$) where B is the Brownian bridge. Then we will accept the hypothesis that the samples are issued from the discrete uniform law if $\sqrt{n} D_n \leq K_\alpha$.

There exist other possible similar tests: for example Anderson-Darling, Cramér–von Mises criterion etc ...
Khi-2

The Khi-2 (or Khi-Squared or χ^2) is a well-known statistical test which can be used to measure the independence of two random variables.

The randomness of the generator can be tested by testing the independence of two disjoint variables issued from the generator. If the generator creates N samples $\{a_1, ..., a_N\}$, then for two permutations σ, θ of $[1, N]$ such as $\sigma(i) \neq \theta(j), \forall i, j = 1, .., N/2$we define $\{a_{\sigma(i)}\} = X$ and $\{a_{\theta(i)}\} = Y$. Therefore the conditions of the χ^2 test for independence of X and Y are respected and we can apply with a given p-value (of 5% for instance)

If $V_1, ..., V_n$ is the set of the n values that can be taken by the generator, then the χ^2 value is :

$\chi^2 = \sum_{s=1}^{n} \frac{(O_s - E_s)^2}{E_s}$
$O_s = \sum_{Y_t = V_s} 1$
$E_s = \frac{\sum_{t=1}^{n} \sum_{Y_t = V_s} 1}{N/2} \sum_{t=1}^{n} \sum_{Y_s = V_t} 1$

O_s and E_s are, respectively, the observed and expected count for the value V_s.

The values V can be {0,1} in case of a bit generator or {0,1,...,255} in case of a byte generator etc ..

The degree of freedom of the χ^2 is, in such a case, $d = (n-1)^2$.
For a bit generator $d = 1$ while for a byte generator, $d = 255^2 = 65,025$ which is too big (at that level the χ^2 law is quasi-equivalent to a normal law and present small interests)

For a nibble generator, $d = (16-1)^2 = 225$ and the χ^2 law can still be used.
Depending on the p-value chosen, the Hypothesis H_0 that the two sets are independent ("null hypothesis") will be chosen depending on the comparison of the value of the computed χ^2 and the reference value for χ^2 at a given p-value for the degree of freedom d.

The Khi-2 test can also be used for "goodness of fit" with a discrete uniform distribution.

5.3 Randomness Tests for Legendre RNG

In order to test the randomness of the Legendre RNG, we apply a certain amount of well-known randomness tests. These tests are used to measure the quality of a random number generator.

We also choose 3 other random generators to be used as a reference against the Legendre RNG. These random generators will be used in the same conditions as the Legendre RNG.

RNG	**FileName**	**Source**	**Comment**
Linux kernel random generator (/dev/random)	linuxrnd.dat	/dev/random	/dev/random takes data from an entropy pool.(see [5])
ESP32 True RNG	esp32hrng.dat	Hardware Rng in ESP32 MCPU	The Hardware based random generator built in the ESP32 processors sold by Espressif. See Appendix A2 for implementation..
Legendre RNG	legendre.dat	Legendre algorithm	See Appendix A1 for implementation.
Sobol RNG	sobol_plus_cpcsp.dat	Sequence created by the Sobol generator	Russian Hardware Generator

The results of each tests is listed in appendix:

	Legendre RNG	**Esp32**	**dev/random**	**Sobol Rng**
Generation	*Completed*	*Completed*	*Completed*	*Completed*
Dieharder	*A6*	*A8*	*A7*	*A9*
NIST	*A5*	*A14*	*A13*	*A10*
ENT	*A3*	*A12*	*A4*	*A11*

5.3.1 Die Harder statistical tests

Presentation DieHarder is a sequel and an improvement to the diehard series of tests developed by George Marsaglia for measuring the quality of the randomness of a RNG.

All the tests of the test suite consist of computing variables generated from various situations in which the random nature of the RNG is assumed. In several of these tests χ^2 and Kolmogorov-Smirnov tests are performed. Most of these tests can be considered as using Monte-Carlo techniques. The description of the tests can be found in [8].

test name	**Sobol**	**Linux**	**ESP32**	**Legendre**
Diehard Birthdays Test	PASSED	PASSED	PASSED	PASSED
Diehard OPERM5 Test	PASSED	PASSED	PASSED	PASSED
Diehard 32x32 Binary Rank Test	PASSED	PASSED	PASSED	PASSED
Diehard 6x8 Binary Rank Test	PASSED	PASSED	PASSED	PASSED
Diehard Bitstream Test	PASSED	PASSED	PASSED	PASSED
Diehard OPSO	PASSED	PASSED	PASSED	PASSED
Diehard OQSO Test	PASSED	PASSED	PASSED	PASSED
Diehard DNA Test	PASSED	PASSED	PASSED	PASSED
Diehard Count the 1s (stream) Test	PASSED	PASSED	PASSED	PASSED
Diehard Count the 1s Test (byte)	PASSED	PASSED	PASSED	PASSED
Diehard Parking Lot Test	PASSED	PASSED	PASSED	PASSED
Diehard Minimum Distance (2d Circle)Test	PASSED	PASSED	PASSED	PASSED
Diehard 3d Sphere (Minimum Distance)Test	PASSED	PASSED	PASSED	PASSED
Diehard Squeeze Test	PASSED	PASSED	PASSED	PASSED
Diehard Sums Test	PASSED	PASSED	PASSED	PASSED
Diehard Runs Test	PASSED	PASSED	PASSED	PASSED
Diehard Craps Test	PASSED	PASSED	PASSED	PASSED
Marsaglia and Tsang GCD Test	PASSED	PASSED	PASSED	FAILED
STS Monobit Test	PASSED	PASSED	PASSED	PASSED
STS Runs Test	PASSED	PASSED	PASSED	PASSED
STS Serial Test (Generalized)	PASSED	PASSED	PASSED	PASSED
RGB Bit Distribution Test	PASSED	PASSED	PASSED	PASSED
RGB Generalized Minimum Distance Test	PASSED	PASSED	PASSED	PASSED
RGB Permutations Test	PASSED	PASSED	PASSED	PASSED
RGB Lagged Sum Test	PASSED	PASSED	PASSED	PASSED
RGB Kolmogorov-Smirnov Test Test	PASSED	PASSED	PASSED	PASSED
Byte Distribution	PASSED	PASSED	PASSED	PASSED
DAB DCT	PASSED	PASSED	PASSED	PASSED
DAB Fill Tree Test	PASSED	PASSED	PASSED	PASSED
DAB Fill Tree 2 Test	PASSED	PASSED	PASSED	PASSED
DAB Monobit 2 Test	PASSED	PASSED	PASSED	PASSED

Results As we see on the summary of the results, the Legendre RNG doesn't behave specially worse than the other reference RNGs.

5.3.2 NIST statistical tests

Presentation The NIST statistical tests (see [3]) are the official suite for FIPS randomness testing.

TestName	Sobol	Linux	ESP32	Legendre
Frequency	PASSED	PASSED	PASSED	PASSED
BlockFrequency	PASSED	PASSED	PASSED	PASSED
CumulativeSums	PASSED	PASSED	PASSED	PASSED
CumulativeSums	PASSED	PASSED	PASSED	PASSED
Runs	PASSED	PASSED	PASSED	PASSED
LongestRun	PASSED	PASSED	PASSED	PASSED
Rank	PASSED	PASSED	PASSED	PASSED
FFT	PASSED	PASSED	PASSED	FAILED
NonOverlappingTemplate	FAILED	FAILED	FAILED	FAILED
Universal	PASSED	PASSED	PASSED	PASSED
ApproximateEntropy	PASSED	PASSED	PASSED	PASSED
RandomExcursions	PASSED	PASSED	FAILED	PASSED
RandomExcursionsVariant	PASSED	PASSED	FAILED	PASSED
Serial	PASSED	PASSED	PASSED	PASSED
LinearComplexity	PASSED	PASSED	PASSED	PASSED

Results Except for the *FFT test*, the Legendre RNG behaves correctly and does not show inferior performances compared to the other random generators.

The FFT test may possibly reveal some issues with the randomness of the Legendre generator but further study would be needed to prove or disprove that fact.

5.3.3 ENT

Presentation ENT performs 'basic' statistical tests. Its complete description can be found in [6] and [7].

1. The 'Entropy' test will compute the Shannon-entropy of the bit stream and see how "close" it is from the maximal Shannon entropy in that context which is 8. This is also interpreted in terms of compression since data appearing as perfectly random noise cannot be compressed at all.
2. The 'Chi-square Test' test will simply perform the χ^2 test of independence between two disjoint sequences S_1 and S_2 taken as a partition of the full sequence S of bits : $S = S_1 \cup S_2$. The p-value which is found for the test is interpreted as a probability that a random sequence would 'exceed' the χ^2 value computed.

3. Basic ‘Monte-carlo’ type tests:

   (a) For N samples of bytes b: $\{b_1, ..., b_N\}$, the mean value $\hat{b} = \frac{1}{N}\sum_{i=1}^{i=N} b_i$ is computed. The expected mean for a uniform distribution is exactly 255/2=127.5.

   (b) Computation of π using monte-carlo method: the test consider the set of points of coordinates $(X_i, Y_i)_{i=1,..,N}$ generated using the RNG, where $0 \leq X \leq 2R$, $0 \leq Y \leq 2R$. If $S(N, R)$ is the amount of points such that $X^2 + Y^2 \leq R^2$, then the expected value of the quantity

$\frac{S(N,R)}{N^2}$ is $\frac{1}{4}\pi$. Hence $\hat{\pi} = 4\frac{S(N,R)}{N^2}$is an estimator of π.

In both cases, how “close" the measured and expected values are is an (empirical) measure of the quality of the randomness.

1. Serial correlation

Under the assumption that the RNG is indeed a time serie of bytes $\{b(t_1), , ..., b(t_N)\}$ following a discrete uniform law *Unif*(0,255) , then since that time serie is stationary and second order stationary, its mean μ is (0+255)/2=127.5 , its variance is $\sigma^2 = 256^2 - 1/12 = 5,461.25$ and we can compute its serial autocorrelation.

The serial covariance is:

$$C_k = E[(b(t_i) - \mu)(b(t_{i+k}) - \mu)]$$

And the serial correlation (autocorrelation) for lag k is:

$\rho_k = C_k/\sigma$

The closer ρ_k is from 0, the more “random" is the generator.

Results
The Legendre Random Generator passes all the tests of the ENT suite.
If we compare its results with the results of the other reference generators, we see that Legendre RNG behaves correctly.

	Sobol	**/dev/random**	**ESP32**	**Legendre**
$\hat{b} - 127.5$	-0,0041000000	0,0140000000	0,0106000000	*0,0070*
$\pi - \hat{\pi}$	0,0003270000	0,0004280810	0,0011928000	*-0.001*
ρ	-0,0000570000	0,000014	-0,000013	*0.0002425*

The computed entropy is 7,999999 for Legendre RNG and concretely identical to the one for the other reference RNGs. No compression is possible.

The χ^2 has the best value among all the other RNG generators tested in the same conditions.

	Sobol	**/dev/random**	**ESP32**	**Legendre**
χ^2	238,67	249,74	281,3	*272.29*

6 Conclusion

As a conclusion, we see that the Legendre RNG exhibits good randomness and passes the statistical tests such as NIST or Dieharder. Legendre RNG may be considered as an acceptable Random number generator.

7 References

[1] Cours D'Arithmétique, Puf, Jean-Pierre Serre
[2] On The Randomness of Legendre and Jacobi Sequences, Ivan Bjerre Damgård
[3] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22
[4] The measures of pseudorandomness and the NIST tests, Merai Rivat & Sarkkozy
[5] /dev/random, Wikipedia article: `https://en.wikipedia.org/wiki//dev/random`
[6] ENT website : `https://www.fourmilab.ch/random/`
[7] ENT man page : Ubuntu Manpage: ent - pseudorandom number sequence test
[8] Dieharder web page : `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`

8 APPENDIX

A1

```
#include "stdafx.h"

typedef long long longint64;

/// <summary>
/// Calculates a Legendre symbol.
/// https://en.wikipedia.org/wiki/Legendre_symbol
/// https://rosettacode.org/wiki/Jacobi_symbol#C++
/// </summary>
/// <param name="a">Cardinal value</param>
/// <param name="p">Odd prime</param>
/// <returns>Symbol value</returns>
static int
calc_legendre_symbol(longint64 n, longint64 k)
{
    _ASSERTE(k > 0 && k % 2 == 1);
```

```
    n %= k;
    longint64 t = 1;
    while (n != 0)
    {
        while (n % 2 == 0)
        {
            n /= 2;
            longint64 r = k % 8;
            if (r == 3 || r == 5)
                t = -t;
        }

        std::swap(n, k);

        if (n % 4 == 3 && k % 4 == 3)
            t = -t;
        n %= k;
    }

    return static_cast<int>(k == 1 ? t : 0);
}


class Random
{
public:

    Random(longint64 p) : _offset(0), _p(p)
    {
    }

    uint32_t nextUint32()
    {
        uint32_t value = 0;

        for (uint32_t i = 0; i < 32; i++)
        {
            value += (uint32_t)((nextBit() ? 1 : 0) « i);
        }

        return value;
    }

private:

    bool nextBit()
    {
        bool value = (calc_legendre_symbol( _offset, _p) == 1);
        _offset++;

        return value;
```

```
    }

    longint64 _offset;
    longint64 _p;
};

/*
void
print_table(std::ostream& out, int kmax, int nmax)
{
    out « "n\\k|";
    for (int k = 0; k <= kmax; ++k)
        out « ' ' « std::setw(2) « k;
    out « "\n—-";
    for (int k = 0; k <= kmax; ++k)
        out « "—";
    out « '\n';
    for (int n = 1; n <= nmax; n += 2) {
        out « std::setw(2) « n « " |";
        for (int k = 0; k <= kmax; ++k)
            out « ' ' « std::setw(2) « calc_legendre_symbol(k, n);
        out « '\n';
    }
}
*/

class OutputRawStream
{
public:
    OutputRawStream(const std::string& outFileName)
    {
#ifdef _WIN32
     errno_t rv = _sopen_s(&_handle, outFileName.c_str(), _O_CREAT | _O_WRONLY | _O_BINARY, _SH_DENYNO,
_S_IWRITE);
        if (rv != 0)
        {
            perror("Unable to open file");
            throw std::logic_error("Unable to open file");
        }
#else
        _handle = open(outFileName.c_str(), O_CREAT | O_RDWR | O_TRUNC, S_IWUSR|S_IRUSR );
        if (-1 == _handle)
        {
            perror("Unable to open file");
            throw std::logic_error("Unable to open file");
        }
#endif // _WIN32
    }

    ~OutputRawStream()
    {
#ifdef _WIN32
```

```
        _close(_handle);
#else
        close(_handle);
#endif // _WIN32
    }

    int handle() const
    {
    return _handle;
    }

    void writeUInt32(uint32_t value)
    {
        this->writeBuffer(&value, sizeof(value));
    }


    void writeByte(unsigned char value)
    {
        this->writeBuffer(&value, sizeof(value));
    }


    void resize(size_t newSize)
    {
    int rv = lseek(_handle, newSize -1, SEEK_SET);
    if(rv < 0)
        throw std::runtime_error("lseek failed");

        int actRd = write(_handle, "", 1);
        if(actRd != 1)
            throw std::runtime_error("write failed");
    }

private:

    void writeBuffer(const void* buff, size_t size)
    {
#ifdef _WIN32
        int rv = _write(_handle, buff, (unsigned int)size);
        if (-1 == rv)
        {
            perror("write error");
            throw std::logic_error("Write error");
        }
#else
        write(_handle, buff, size);
#endif // _WIN32
    }

    int _handle;
```

```
};



class MemoryMap
{
public:
    MemoryMap(int handle, size_t len, int prot, int flags, off_t offset) : _size(len)
    {
    _addr = mmap(NULL, len, prot, flags, handle, offset);
    if(MAP_FAILED == _addr)
    {
        perror("Mapping failed");
        throw std::logic_error("Mapping failed");
    }
    }


    ~MemoryMap() noexcept
    {
    munmap(_addr, _size);
    }


    void* addr()
    {
    return _addr;
    }

private:

    void* _addr;
    size_t _size;
};




int
main(int argc, char *argv[])
{

    //const std::string str = "594348534879";
    //unsigned long long v = std::stoull(str);
    //size_t z = sizeof(unsigned long long);

    std::cout « "Size of longint64: " « sizeof(longint64) « std::endl;

    if (argc < 5)
    {
        std::cerr « "\n\tUsage: legendreplus.exe output-file-name num-of-random-values odd-prime fmt" « std::endl;
```

```
        std::cerr « "\n\t Where 'fmt' is output format. 1- binary, 0 - text" « std::endl;
        return 5;
    }

    size_t    maxNum = atoi(argv[2]);
    longint64 prime = std::stoll(argv[3]);

    Random rnd(prime);



    OutputRawStream outStm(argv[1]);
    size_t outStmSize = sizeof(uint32_t) * maxNum;

    outStm.resize(outStmSize);
    std::cout « "New file size is " « outStmSize « std::endl;

    MemoryMap memmap(outStm.handle(), outStmSize, PROT_WRITE, MAP_SHARED, 0);
    uint32_t* mapAddr = (uint32_t*)memmap.addr();

    if (argv[4][0] == '1')
    {

        for (size_t k = 2; k < maxNum; k++)
        {
            uint32_t nextval = rnd.nextUint32();
            *mapAddr = nextval;
            mapAddr++;

        if(k % 1000 == 0)
        {
        std::cout « k « " values of " « maxNum « " were created" « std::endl;
        }
        }
    }
    else
    {
        for (size_t k = 2; k < maxNum; k++)
        {
            int lsym = calc_legendre_symbol(k, prime);
            if(lsym == 1)
            {
                outStm.writeByte('1');
            }
            else
                outStm.writeByte('0');
        }
    }
    return 0;
}
```

A2

```
/*
   Hardware random generator. Based on ESP32.
   ESP32 has a HRNG on board. The generator
   works, if WiFi or BT is on.

   Random data will be stored on the SD card.

   Peter A. Smirnoff , SCD(C), 2022
*/
#include <WiFi.h>
#include <FS.h>
#include <SD.h>
#include <SPI.h>


bool failed = false;

File rawfile;

void
createDir(fs::FS &fs, const char * path) {
  Serial.printf("Creating Dir: %s\n", path);
  if (fs.mkdir(path)) {
    Serial.println("Dir created");
  } else {
    Serial.println("mkdir failed");
  }
}



void
removeDir(fs::FS &fs, const char * path) {
  Serial.printf("Removing Dir: %s\n", path);
  if (fs.rmdir(path)) {
    Serial.println("Dir removed");
  } else {
    Serial.println("rmdir failed");
  }
}

void
setup()
{
  Serial.begin(115200);
  //
  // It's very important to switch on a WiFi or BlueTooth,
  // or RNG will be in SOFTWARE...
  //
  Serial.print("Switching on WiFi...");
  WiFi.mode(WIFI_STA);
```

```
  WiFi.disconnect();
  delay(100);
  Serial.println("DONE!");
  //
  // Check SD-card subsystem readiness
  //
  if (!SD.begin())
  {
    Serial.println("ERROR: Card Mount Failed");
    failed = true;
    return;
  }
  //
  // Check card type. We need
  //
  uint8_t cardType = SD.cardType();
  if (cardType != CARD_SDHC)
  {
    failed = true;
    Serial.println("No SD card or unproper card type attached ");
    return;
  }

  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("SD Card Size: %lluMB\n", cardSize);

  Serial.println("Erasing pre-existing directory...");
  removeDir(SD, "/rndfiles");

  Serial.println("Creating directory...");
  createDir(SD, "/rndfiles");

  rawfile = SD.open("/rndfiles/esp32hdng.dat", FILE_WRITE);
  Serial.println("== Alles fertig! ==");
}


uint32_t count = 0;

void
loop()
{
  if (count > 10000000) // amount of data to be generated
  {
    Serial.println("Limit is reached, exiting...");
    rawfile.close();
    delay(5000);
    return;
  }

  uint32_t value = esp_random(); // 4 octets of random data
```

```
  //Serial.print("Value = ");
  //Serial.println(value, HEX);
  //
  // Write data
  //
  rawfile.write((const byte*)&value, sizeof(value));
  //delay(100);
  //Serial.print("Data portion was written: ");
  //Serial.println(count);
  count++;
}
```

A3
ENT+LegendreRNG
[peter@jisa-serv data]$ /opt/ent/bin/ent ./legendre.dat
Entropy = 7.999995 bits per byte.

Optimum compression would reduce the size
of this 40000000 byte file by 0 percent.

Chi square distribution for 40000000 samples is 272.29, and randomly
would exceed this value 21.82 percent of the times.

Arithmetic mean value of data bytes is 127.5070 (127.5 = random).
Monte Carlo value for Pi is 3.142827914 (error 0.04 percent).
Serial correlation coefficient is 0.000242 (totally uncorrelated = 0.0).

A4
ENT+dev/random
Entropy = 7.999999 bits per byte.

Optimum compression would reduce the size
of this 125829120 byte file by 0 percent.

Chi square distribution for 125829120 samples is 249.74, and randomly
would exceed this value 58.13 percent of the times.

Arithmetic mean value of data bytes is 127.5140 (127.5 = random).
Monte Carlo value for Pi is 3.141161919 (error 0.01 percent).
Serial correlation coefficient is 0.000014 (totally uncorrelated = 0.0).
.
A5
LegendreRNG+NIST
Samples: legendre.dat 120 000 000 bytes

```
Passphrase for key "imported-openssh-key":
Last login: Sat Oct  1 20:10:50 2022 from 134.119.194.82

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 2 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-38-140 ~]$ cd
.cache/   legendre/ .ssh/
[ec2-user@ip-172-31-38-140 ~]$ cd legendre/
[ec2-user@ip-172-31-38-140 legendre]$ dir
code.7z  code.zip  legendre  legendreRNG-main  legendreRNG-main.zip  sts-2.1.2  sts.zip
[ec2-user@ip-172-31-38-140 legendre]$ cd sts-2.1.2/
[ec2-user@ip-172-31-38-140 sts-2.1.2]$ cd sts-2.1.2/
[ec2-user@ip-172-31-38-140 sts-2.1.2]$ dir
assess  data  experiments  include  makefile  obj  results  src  stats  templates
[ec2-user@ip-172-31-38-140 sts-2.1.2]$ ./assess 39999744
```

ec2-user@ip-172-31-38-140:~/legendre/sts-2.1.2/sts-2.1.2

```
             User Prescribed Input File: /home/ec2-user/legendre/legendreRNG-

             S T A T I S T I C A L   T E S T S
             _________________________________

  [01] Frequency                       [02] Block Frequency
  [03] Cumulative Sums                 [04] Runs
  [05] Longest Run of Ones             [06] Rank
  [07] Discrete Fourier Transform      [08] Nonperiodic Template Matchings
  [09] Overlapping Template Matchings  [10] Universal Statistical
  [11] Approximate Entropy             [12] Random Excursions
  [13] Random Excursions Variant       [14] Serial
  [15] Linear Complexity

       INSTRUCTIONS
          Enter 0 if you DO NOT want to apply all of the
          statistical tests to each sequence and 1 if you DO.

 Enter Choice: 1

      P a r a m e t e r   A d j u s t m e n t s
      ----------------------------------------
  [1] Block Frequency Test - block length(M):         128
  [2] NonOverlapping Template Test - block length(m): 9
  [3] Overlapping Template Test - block length(m):    9
  [4] Approximate Entropy Test - block length(m):     10
  [5] Serial Test - block length(m):                  16
  [6] Linear Complexity Test - block length(M):       500

 Select Test (0 to continue): 0

 How many bitstreams? 1

 Input File Format:
  [0] ASCII - A sequence of ASCII 0's and 1's
  [1] Binary - Each byte in data file contains 8 bits of data

 Select input mode:  1

   Statistical Testing In Progress.........
```

———————————————————————————————

———————————————————————————————

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

———————————————————————————————

generator is </home/peter/projects/data/legendre.dat>

———————————————————————————————

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
0	0	2	4	0	1	0	1	1	1	0.122325	10/10	Frequency
0	0	2	0	3	0	2	2	1	0	0.213309	10/10	BlockFrequency
0	0	0	3	1	3	1	0	0	2	0.122325	10/10	CumulativeSums
0	0	2	2	2	2	1	0	0	1	0.534146	10/10	CumulativeSums
0	0	1	7	0	0	1	0	1	0	0.000003 *	10/10	Runs
0	0	3	0	1	0	1	4	1	0	0.035174	10/10	LongestRun
1	1	0	2	1	1	2	1	0	1	0.911413	10/10	Rank
10	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0.000000 ***	**0/10 ***	**FFT**
0	4	0	0	0	1	2	2	1	0	0.066882	10/10	NonOverlappingTemplate
1	1	0	1	0	1	1	3	1	1	0.739918	10/10	NonOverlappingTemplate
2	0	1	3	0	1	3	0	0	0	0.122325	9/10	NonOverlappingTemplate
0	3	1	0	3	0	1	2	0	0	0.122325	10/10	NonOverlappingTemplate
2	2	0	1	1	0	0	1	0	3	0.350485	10/10	NonOverlappingTemplate
1	2	1	1	2	0	0	1	0	2	0.739918	10/10	NonOverlappingTemplate
2	0	3	1	1	1	0	0	1	1	0.534146	10/10	NonOverlappingTemplate
2	1	1	3	1	0	0	0	0	2	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	1	2	1	0	1	3	0.534146	10/10	NonOverlappingTemplate
0	1	2	0	0	3	1	1	1	1	0.534146	10/10	NonOverlappingTemplate
1	1	0	0	2	0	1	3	1	1	0.534146	10/10	NonOverlappingTemplate
0	0	0	1	2	2	1	1	1	2	0.739918	10/10	NonOverlappingTemplate
1	0	2	0	0	1	1	2	2	1	0.739918	10/10	NonOverlappingTemplate
0	3	0	1	0	2	2	2	0	0	0.213309	10/10	NonOverlappingTemplate
0	1	0	0	2	1	1	3	2	0	0.350485	10/10	NonOverlappingTemplate
2	1	1	0	1	1	1	1	2	0	0.911413	10/10	NonOverlappingTemplate
1	0	0	0	0	3	0	2	0	4	0.017912	10/10	NonOverlappingTemplate
2	2	1	0	1	0	3	0	0	1	0.350485	10/10	NonOverlappingTemplate
0	1	0	1	1	3	1	1	1	1	0.739918	10/10	NonOverlappingTemplate
0	2	0	1	0	2	0	1	2	2	0.534146	10/10	NonOverlappingTemplate
3	0	2	1	1	0	1	0	0	2	0.350485	9/10	NonOverlappingTemplate
0	2	1	1	2	0	0	1	1	2	0.739918	10/10	NonOverlappingTemplate
1	0	0	2	0	2	1	1	2	1	0.739918	10/10	NonOverlappingTemplate
0	1	0	4	1	0	0	1	1	2	0.122325	10/10	NonOverlappingTemplate
0	1	0	2	3	3	0	1	0	0	0.122325	10/10	NonOverlappingTemplate
3	1	0	0	0	0	1	1	3	1	0.213309	10/10	NonOverlappingTemplate
2	0	3	0	0	1	1	1	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	1	0	1	1	2	2	2	0	0.739918	10/10	NonOverlappingTemplate
0	1	0	0	1	0	4	0	4	0	0.004301	10/10	NonOverlappingTemplate
0	2	1	0	0	4	1	0	2	0	0.066882	10/10	NonOverlappingTemplate
3	1	1	1	0	1	1	2	0	0	0.534146	10/10	NonOverlappingTemplate

1	1	2	1	0	2	2	0	1	0	0.739918	10/10	NonOverlappingTemplate
1	0	2	2	0	3	0	0	2	0	0.213309	10/10	NonOverlappingTemplate
1	3	0	1	0	2	1	0	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	1	1	1	3	1	2	0	0.534146	10/10	NonOverlappingTemplate
0	0	1	1	0	4	1	2	0	1	0.122325	10/10	NonOverlappingTemplate
1	2	1	1	0	1	0	1	2	1	0.911413	10/10	NonOverlappingTemplate
2	1	1	0	0	1	1	0	2	2	0.739918	9/10	NonOverlappingTemplate
2	0	3	2	2	1	0	0	0	0	0.213309	10/10	NonOverlappingTemplate
1	0	2	1	3	0	2	1	0	0	0.350485	10/10	NonOverlappingTemplate
3	0	0	1	2	2	0	0	1	1	0.350485	10/10	NonOverlappingTemplate
1	2	1	0	3	1	0	1	1	0	0.534146	10/10	NonOverlappingTemplate
0	2	0	1	1	4	0	0	2	0	0.066882	10/10	NonOverlappingTemplate
2	1	0	1	1	3	1	0	0	1	0.534146	10/10	NonOverlappingTemplate
0	0	0	0	3	4	1	0	1	1	0.035174	10/10	NonOverlappingTemplate
1	0	2	2	0	1	0	1	3	0	0.350485	10/10	NonOverlappingTemplate
2	0	1	2	1	0	0	1	1	2	0.739918	9/10	NonOverlappingTemplate
2	0	3	1	2	0	0	0	0	2	0.213309	10/10	NonOverlappingTemplate
0	1	0	1	1	1	2	1	3	0	0.534146	10/10	NonOverlappingTemplate
4	2	0	1	0	1	1	0	1	0	0.122325	10/10	NonOverlappingTemplate
0	2	0	2	3	1	0	0	2	0	0.213309	10/10	NonOverlappingTemplate
0	0	2	0	0	2	2	0	1	3	0.213309	10/10	NonOverlappingTemplate
1	4	0	1	0	1	0	0	2	1	0.122325	10/10	NonOverlappingTemplate
1	0	2	0	1	1	1	3	1	0	0.534146	10/10	NonOverlappingTemplate
2	1	2	0	1	0	1	0	2	1	0.739918	9/10	NonOverlappingTemplate
1	1	0	2	1	0	3	1	0	1	0.534146	10/10	NonOverlappingTemplate
2	1	1	0	0	1	1	0	4	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	2	0	0	2	2	0	3	0.213309	10/10	NonOverlappingTemplate
0	0	1	0	1	0	1	2	3	2	0.350485	10/10	NonOverlappingTemplate
0	2	1	3	0	2	0	1	1	0	0.350485	10/10	NonOverlappingTemplate
2	1	0	0	2	2	1	0	0	2	0.534146	9/10	NonOverlappingTemplate
1	3	1	0	1	2	2	0	0	0	0.350485	9/10	NonOverlappingTemplate
2	0	1	1	1	3	1	1	0	0	0.534146	10/10	NonOverlappingTemplate
0	1	1	3	1	1	0	1	1	1	0.739918	10/10	NonOverlappingTemplate
1	3	0	0	2	1	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
1	1	3	0	1	2	0	1	0	1	0.534146	10/10	NonOverlappingTemplate
0	2	0	2	0	4	0	1	1	0	0.066882	10/10	NonOverlappingTemplate
3	0	1	0	1	1	0	2	0	2	0.350485	10/10	NonOverlappingTemplate
1	1	1	1	0	3	0	1	1	1	0.739918	10/10	NonOverlappingTemplate
0	4	2	1	1	0	0	0	1	1	0.122325	10/10	NonOverlappingTemplate
0	1	0	2	0	1	2	1	2	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	2	0	1	2	2	0	0	0.739918	10/10	NonOverlappingTemplate
1	0	1	1	2	0	1	2	2	0	0.739918	10/10	NonOverlappingTemplate
0	0	3	1	0	3	2	0	0	1	0.122325	10/10	NonOverlappingTemplate
0	4	0	0	0	1	2	2	1	0	0.066882	10/10	NonOverlappingTemplate
2	0	2	1	1	4	0	0	0	0	0.066882	10/10	NonOverlappingTemplate
0	2	1	1	1	0	4	0	1	0	0.122325	10/10	NonOverlappingTemplate
1	2	1	2	2	0	0	1	1	0	0.739918	10/10	NonOverlappingTemplate
0	1	0	1	1	2	3	0	0	2	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	3	1	2	0	0	2	0.350485	10/10	NonOverlappingTemplate
3	1	0	1	1	1	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	1	0	1	2	0	2	1	1	2	0.739918	10/10	NonOverlappingTemplate

0	1	2	0	0	0	1	1	1	4	0.122325	10/10	NonOverlappingTemplate
1	1	2	1	1	1	1	0	1	1	0.991468	10/10	NonOverlappingTemplate
1	1	1	1	1	2	1	1	0	1	0.991468	10/10	NonOverlappingTemplate
3	1	0	0	1	2	2	0	1	0	0.350485	10/10	NonOverlappingTemplate
2	1	2	0	1	2	1	0	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	1	1	3	1	0	1	0	0.739918	10/10	NonOverlappingTemplate
2	0	1	0	1	0	2	2	1	1	0.739918	10/10	NonOverlappingTemplate
1	0	1	0	2	1	1	2	1	1	0.911413	10/10	NonOverlappingTemplate
1	0	0	1	1	2	4	1	0	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	3	2	2	0	0	1	1	0.350485	10/10	NonOverlappingTemplate
0	1	1	1	0	2	1	2	0	2	0.739918	10/10	NonOverlappingTemplate
0	1	0	0	2	4	0	1	2	0	0.066882	10/10	NonOverlappingTemplate
2	1	1	2	0	1	3	0	0	0	0.350485	10/10	NonOverlappingTemplate
0	1	2	0	0	2	3	1	0	1	0.350485	10/10	NonOverlappingTemplate
1	1	1	0	0	2	0	3	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	2	2	1	0	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	0	0	1	1	2	2	2	0	0.739918	10/10	NonOverlappingTemplate
1	1	2	0	1	0	1	2	1	1	0.911413	10/10	NonOverlappingTemplate
2	2	1	1	1	0	2	0	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	0	1	0	0	2	2	3	0	0.350485	10/10	NonOverlappingTemplate
0	0	1	2	1	1	0	2	2	1	0.739918	10/10	NonOverlappingTemplate
2	0	2	0	3	1	2	0	0	0	0.213309	10/10	NonOverlappingTemplate
0	0	0	2	2	0	2	1	1	2	0.534146	10/10	NonOverlappingTemplate
2	2	0	1	1	0	1	0	1	2	0.739918	10/10	NonOverlappingTemplate
2	3	2	1	0	0	0	0	2	0	0.213309	9/10	NonOverlappingTemplate
1	0	2	1	1	1	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
0	0	1	2	3	0	2	2	0	0	0.213309	10/10	NonOverlappingTemplate
1	3	0	1	2	1	0	1	0	1	0.534146	10/10	NonOverlappingTemplate
2	4	0	1	2	0	0	1	0	0	0.066882	10/10	NonOverlappingTemplate
2	3	2	0	1	0	0	2	0	0	0.213309	10/10	NonOverlappingTemplate
1	2	2	0	0	1	0	1	2	1	0.739918	10/10	NonOverlappingTemplate
2	1	0	1	2	1	0	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	2	1	1	2	1	1	2	0	0	0.739918	10/10	NonOverlappingTemplate
0	2	1	0	1	0	0	1	5	0	0.008879	10/10	NonOverlappingTemplate
2	1	0	0	1	2	2	0	0	2	0.534146	10/10	NonOverlappingTemplate
0	1	1	2	0	2	2	0	1	1	0.739918	10/10	NonOverlappingTemplate
1	4	0	0	1	2	1	0	0	1	0.122325	10/10	NonOverlappingTemplate
0	0	3	0	2	0	0	2	2	1	0.213309	10/10	NonOverlappingTemplate
0	0	0	1	2	3	1	1	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	2	2	1	0	3	1	0	0.350485	10/10	NonOverlappingTemplate
1	1	0	2	2	1	2	0	0	1	0.739918	10/10	NonOverlappingTemplate
2	3	1	0	0	2	0	0	1	1	0.350485	10/10	NonOverlappingTemplate
0	1	3	1	2	0	0	0	1	2	0.350485	10/10	NonOverlappingTemplate
1	0	0	1	0	1	1	0	2	4	0.122325	10/10	NonOverlappingTemplate
0	2	2	0	0	0	2	2	2	0	0.350485	10/10	NonOverlappingTemplate
2	2	1	1	2	1	1	0	0	0	0.739918	10/10	NonOverlappingTemplate
2	0	0	1	1	4	1	1	0	0	0.122325	10/10	NonOverlappingTemplate
0	0	2	0	1	1	2	1	0	3	0.350485	10/10	NonOverlappingTemplate
1	3	0	0	1	0	1	3	1	0	0.213309	10/10	NonOverlappingTemplate
0	0	2	3	1	2	0	0	2	0	0.213309	10/10	NonOverlappingTemplate
2	0	2	1	1	1	2	1	0	0	0.739918	10/10	NonOverlappingTemplate

0	2	0	1	0	1	1	1	2	2	0.739918	10/10	NonOverlappingTemplate
0	1	2	0	2	2	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	2	2	1	1	0	0	2	1	1	0.739918	10/10	NonOverlappingTemplate
2	0	0	2	0	0	1	2	1	2	0.534146	10/10	NonOverlappingTemplate
1	2	0	1	2	1	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
0	0	0	0	1	2	3	1	1	2	0.350485	10/10	NonOverlappingTemplate
4	2	0	1	0	1	0	2	0	0	0.066882	10/10	NonOverlappingTemplate
2	0	3	2	1	1	1	0	0	0	0.350485	9/10	NonOverlappingTemplate
1	0	1	4	0	0	2	0	0	2	0.066882	10/10	NonOverlappingTemplate
1	2	2	1	1	2	0	1	0	0	0.739918	10/10	NonOverlappingTemplate
0	2	0	3	0	1	2	1	0	1	0.350485	10/10	NonOverlappingTemplate
3	1	1	2	1	1	0	1	0	0	0.534146	10/10	NonOverlappingTemplate
1	1	1	0	0	1	2	1	1	2	0.911413	10/10	NonOverlappingTemplate
1	1	0	0	0	1	0	2	3	2	0.350485	10/10	NonOverlappingTemplate
0	0	3	1	0	3	2	0	0	1	0.122325	10/10	NonOverlappingTemplate
7	2	0	1	0	0	0	0	0	0	0.000001 *	7/10	* OverlappingTemplate
1	2	1	1	2	1	1	0	1	0	0.911413	10/10	Universal
1	1	1	2	0	1	1	1	0	2	0.911413	10/10	ApproximateEntropy
3	0	0	3	0	2	1	1	0	0	0.122325	10/10	RandomExcursions
0	3	1	1	2	0	1	1	1	0	0.534146	10/10	RandomExcursions
3	2	0	0	0	1	1	0	3	0	0.122325	10/10	RandomExcursions
3	2	2	0	0	0	0	2	0	1	0.213309	10/10	RandomExcursions
3	0	1	1	0	1	2	1	1	0	0.534146	8/10	RandomExcursions
1	1	0	1	1	1	1	2	2	0	0.911413	9/10	RandomExcursions
2	0	2	0	1	2	0	1	1	1	0.739918	9/10	RandomExcursions
4	1	0	0	1	1	0	0	3	0	0.035174	9/10	RandomExcursions
1	0	0	1	1	0	2	1	1	3	0.534146	10/10	RandomExcursionsVariant
1	0	0	0	2	1	2	1	1	2	0.739918	10/10	RandomExcursionsVariant
1	0	0	1	2	1	2	2	1	0	0.739918	10/10	RandomExcursionsVariant
1	0	1	1	0	1	4	1	0	1	0.213309	10/10	RandomExcursionsVariant
0	3	0	0	1	2	2	0	1	1	0.350485	10/10	RandomExcursionsVariant
1	0	3	0	3	0	2	0	1	0	0.122325	10/10	RandomExcursionsVariant
1	2	0	1	2	0	1	0	2	1	0.739918	10/10	RandomExcursionsVariant
2	0	0	1	0	0	3	0	3	1	0.122325	10/10	RandomExcursionsVariant
0	2	0	1	1	0	2	1	2	1	0.739918	10/10	RandomExcursionsVariant
2	0	1	0	3	1	0	2	0	1	0.350485	8/10	RandomExcursionsVariant
3	0	1	1	1	1	3	0	0	0	0.213309	9/10	RandomExcursionsVariant
2	1	1	2	0	3	0	0	1	0	0.350485	9/10	RandomExcursionsVariant
3	0	0	3	1	2	1	0	0	0	0.122325	10/10	RandomExcursionsVariant
2	2	0	2	2	1	1	0	0	0	0.534146	9/10	RandomExcursionsVariant
2	2	2	0	1	2	0	0	1	0	0.534146	9/10	RandomExcursionsVariant
3	2	0	1	1	1	2	0	0	0	0.350485	9/10	RandomExcursionsVariant
3	1	1	0	2	1	0	1	0	1	0.534146	9/10	RandomExcursionsVariant
2	2	0	1	0	2	0	2	0	1	0.534146	10/10	RandomExcursionsVariant
2	0	0	2	2	1	1	1	0	1	0.739918	9/10	Serial
2	1	0	2	1	1	3	0	0	0	0.350485	9/10	Serial
0	1	1	2	0	2	0	0	2	2	0.534146	10/10	LinearComplexity

- -

The minimum pass rate for each statistical test with the exception of the

random excursion (variant) test is approximately = 8 for a sample size = 10 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately = 8 for a sample size = 10 binary sequences.

For further guidelines construct a probability table using the MAPLE program provided in the addendum section of the documentation.
- -

A6
LegendreRNG+DIEHARDER
Samples: legendre.dat, 2000000000 bytes

```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
file_input_raw|              ./legendre-2Gb.dat| 3.48e+07 |
#=============================================================================#
        test_name   |ntup| tsamples |psamples| p-value |Assessment
#=============================================================================#
   diehard_birthdays|   0|       100|     100|0.72390043| PASSED
      diehard_operm5|   0|   1000000|     100|0.81484841| PASSED
  diehard_rank_32x32|   0|     40000|     100|0.73203913| PASSED
    diehard_rank_6x8|   0|    100000|     100|0.98664859| PASSED
   diehard_bitstream|   0|   2097152|     100|0.46898427| PASSED
        diehard_opso|   0|   2097152|     100|0.66349780| PASSED
        diehard_oqso|   0|   2097152|     100|0.27759975| PASSED
         diehard_dna|   0|   2097152|     100|0.79286040| PASSED
diehard_count_1s_str|   0|    256000|     100|0.10412903| PASSED
diehard_count_1s_byt|   0|    256000|     100|0.56927236| PASSED
 diehard_parking_lot|   0|     12000|     100|0.98727196| PASSED
    diehard_2dsphere|   2|      8000|     100|0.07945198| PASSED
    diehard_3dsphere|   3|      4000|     100|0.61359999| PASSED
     diehard_squeeze|   0|    100000|     100|0.69724219| PASSED
        diehard_sums|   0|       100|     100|0.01769197| PASSED
        diehard_runs|   0|    100000|     100|0.32849425| PASSED
        diehard_runs|   0|    100000|     100|0.99738089|  WEAK
       diehard_craps|   0|    200000|     100|0.87161743| PASSED
       diehard_craps|   0|    200000|     100|0.89023751| PASSED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.00000023| FAILED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.04633785| PASSED
         sts_monobit|   1|    100000|     100|0.79241023| PASSED
            sts_runs|   2|    100000|     100|0.54505484| PASSED
          sts_serial|   1|    100000|     100|0.92912390| PASSED
          sts_serial|   2|    100000|     100|0.60302836| PASSED
          sts_serial|   3|    100000|     100|0.07897488| PASSED
          sts_serial|   3|    100000|     100|0.03048894| PASSED
          sts_serial|   4|    100000|     100|0.32720655| PASSED
          sts_serial|   4|    100000|     100|0.82654388| PASSED
          sts_serial|   5|    100000|     100|0.19953985| PASSED
```

sts_serial	5	100000	100	0.02535283	PASSED
sts_serial	6	100000	100	0.46529268	PASSED
sts_serial	6	100000	100	0.04649420	PASSED
sts_serial	7	100000	100	0.99913674	WEAK
sts_serial	7	100000	100	0.40442450	PASSED
sts_serial	8	100000	100	0.92860832	PASSED
sts_serial	8	100000	100	0.63753206	PASSED
sts_serial	9	100000	100	0.49781344	PASSED
sts_serial	9	100000	100	0.54703966	PASSED
sts_serial	10	100000	100	0.64553474	PASSED
sts_serial	10	100000	100	0.29341895	PASSED
sts_serial	11	100000	100	0.15918599	PASSED
sts_serial	11	100000	100	0.08598391	PASSED
sts_serial	12	100000	100	0.94311484	PASSED
sts_serial	12	100000	100	0.99595566	WEAK
sts_serial	13	100000	100	0.70208054	PASSED
sts_serial	13	100000	100	0.77419000	PASSED
sts_serial	14	100000	100	0.45192818	PASSED
sts_serial	14	100000	100	0.29894313	PASSED
sts_serial	15	100000	100	0.24138564	PASSED
sts_serial	15	100000	100	0.08921693	PASSED
sts_serial	16	100000	100	0.01956695	PASSED
sts_serial	16	100000	100	0.02233211	PASSED
rgb_bitdist	1	100000	100	0.50807322	PASSED
rgb_bitdist	2	100000	100	0.00808459	PASSED
rgb_bitdist	3	100000	100	0.71954254	PASSED
rgb_bitdist	4	100000	100	0.26875571	PASSED
rgb_bitdist	5	100000	100	0.29767707	PASSED
rgb_bitdist	6	100000	100	0.46124964	PASSED
rgb_bitdist	7	100000	100	0.70361974	PASSED
rgb_bitdist	8	100000	100	0.00380899	WEAK
rgb_bitdist	9	100000	100	0.15669307	PASSED
rgb_bitdist	10	100000	100	0.34847030	PASSED
rgb_bitdist	11	100000	100	0.44556904	PASSED
rgb_bitdist	12	100000	100	0.04665157	PASSED
rgb_minimum_distance	2	10000	1000	0.81106227	PASSED
rgb_minimum_distance	3	10000	1000	0.80619022	PASSED
rgb_minimum_distance	4	10000	1000	0.95825655	PASSED
rgb_minimum_distance	5	10000	1000	0.55449360	PASSED
rgb_permutations	2	100000	100	0.91322641	PASSED
rgb_permutations	3	100000	100	0.37634516	PASSED
rgb_permutations	4	100000	100	0.94564776	PASSED
rgb_permutations	5	100000	100	0.95414609	PASSED
rgb_lagged_sum	0	1000000	100	0.46937023	PASSED
rgb_lagged_sum	1	1000000	100	0.28632876	PASSED
rgb_lagged_sum	2	1000000	100	0.75552971	PASSED
rgb_lagged_sum	3	1000000	100	0.85493323	PASSED
rgb_lagged_sum	4	1000000	100	0.15596665	PASSED
rgb_lagged_sum	5	1000000	100	0.18569191	PASSED
rgb_lagged_sum	6	1000000	100	0.99338796	PASSED
rgb_lagged_sum	7	1000000	100	0.22716169	PASSED

```
      rgb_lagged_sum|   8|   1000000|     100|0.71634529| PASSED
      rgb_lagged_sum|   9|   1000000|     100|0.39340548| PASSED
      rgb_lagged_sum|  10|   1000000|     100|0.38789190| PASSED
      rgb_lagged_sum|  11|   1000000|     100|0.67755494| PASSED
      rgb_lagged_sum|  12|   1000000|     100|0.58761985| PASSED
      rgb_lagged_sum|  13|   1000000|     100|0.33860602| PASSED
      rgb_lagged_sum|  14|   1000000|     100|0.37185325| PASSED
      rgb_lagged_sum|  15|   1000000|     100|0.01214439| PASSED
      rgb_lagged_sum|  16|   1000000|     100|0.70892160| PASSED
      rgb_lagged_sum|  17|   1000000|     100|0.38635683| PASSED
      rgb_lagged_sum|  18|   1000000|     100|0.30756214| PASSED
      rgb_lagged_sum|  19|   1000000|     100|0.00745131| PASSED
      rgb_lagged_sum|  20|   1000000|     100|0.57039806| PASSED
      rgb_lagged_sum|  21|   1000000|     100|0.03180119| PASSED
      rgb_lagged_sum|  22|   1000000|     100|0.98929861| PASSED
      rgb_lagged_sum|  23|   1000000|     100|0.04033254| PASSED
      rgb_lagged_sum|  24|   1000000|     100|0.00000000| FAILED
      rgb_lagged_sum|  25|   1000000|     100|0.01630303| PASSED
      rgb_lagged_sum|  26|   1000000|     100|0.99407761| PASSED
      rgb_lagged_sum|  27|   1000000|     100|0.48218461| PASSED
      rgb_lagged_sum|  28|   1000000|     100|0.15747137| PASSED
      rgb_lagged_sum|  29|   1000000|     100|0.04838112| PASSED
      rgb_lagged_sum|  30|   1000000|     100|0.76025218| PASSED
      rgb_lagged_sum|  31|   1000000|     100|0.03866215| PASSED
      rgb_lagged_sum|  32|   1000000|     100|0.96364348| PASSED
  rgb_kstest_test|   0|     10000|    1000|0.60101019| PASSED
  dab_bytedistrib|   0| 51200000|       1|0.13533585| PASSED
          dab_dct| 256|     50000|       1|0.82254279| PASSED
Preparing to run test 207. ntuple = 0
       dab_filltree|  32| 15000000|       1|0.05999639| PASSED
       dab_filltree|  32| 15000000|       1|0.97067570| PASSED
Preparing to run test 208. ntuple = 0
      dab_filltree2|   0|   5000000|       1|0.20556592| PASSED
      dab_filltree2|   1|   5000000|       1|0.16996793| PASSED
Preparing to run test 209. ntuple = 0
       dab_monobit2|  12|  65000000|       1|0.12824613| PASSED
```

A7
Rng source: Linux /dev/random
Test suite : DieHarder
Samples amount: 2000 000 000 octets
Results are below:

```
#=============================================================================
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================
   rng_name    |           filename             |rands/second|
file_input_raw|      ./linuxrnd-2Gb-exactly.dat| 3.31e+07 |
#=============================================================================
        test_name   |ntup| tsamples |psamples| p-value |Assessment
#=============================================================================
```

diehard_birthdays| 0| 100| 100|0.22427196| PASSED
diehard_operm5| 0| 1000000| 100|0.63757741| PASSED
diehard_rank_32x32| 0| 40000| 100|0.95981163| PASSED
diehard_rank_6x8| 0| 100000| 100|0.31198460| PASSED
diehard_bitstream| 0| 2097152| 100|0.60771534| PASSED
diehard_opso| 0| 2097152| 100|0.76781669| PASSED
diehard_oqso| 0| 2097152| 100|0.84563285| PASSED
diehard_dna| 0| 2097152| 100|0.01922252| PASSED
diehard_count_1s_str| 0| 256000| 100|0.20098076| PASSED
diehard_count_1s_byt| 0| 256000| 100|0.17039138| PASSED
diehard_parking_lot| 0| 12000| 100|0.99430846| PASSED
diehard_2dsphere| 2| 8000| 100|0.35115398| PASSED
diehard_3dsphere| 3| 4000| 100|0.73195045| PASSED
diehard_squeeze| 0| 100000| 100|0.73969241| PASSED
diehard_sums| 0| 100| 100|0.12325827| PASSED
diehard_runs| 0| 100000| 100|0.97674640| PASSED
diehard_runs| 0| 100000| 100|0.60592037| PASSED
diehard_craps| 0| 200000| 100|0.70715861| PASSED
diehard_craps| 0| 200000| 100|0.93273632| PASSED
marsaglia_tsang_gcd| 0| 10000000| 100|0.24162026| PASSED
marsaglia_tsang_gcd| 0| 10000000| 100|0.05414982| PASSED
sts_monobit| 1| 100000| 100|0.42869783| PASSED
sts_runs| 2| 100000| 100|0.96625227| PASSED
sts_serial| 1| 100000| 100|0.21061188| PASSED
sts_serial| 2| 100000| 100|0.10861589| PASSED
sts_serial| 3| 100000| 100|0.60895261| PASSED
sts_serial| 3| 100000| 100|0.83762291| PASSED
sts_serial| 4| 100000| 100|0.53723490| PASSED
sts_serial| 4| 100000| 100|0.23132511| PASSED
sts_serial| 5| 100000| 100|0.57974825| PASSED
sts_serial| 5| 100000| 100|0.64682304| PASSED
sts_serial| 6| 100000| 100|0.38762980| PASSED
sts_serial| 6| 100000| 100|0.15504094| PASSED
sts_serial| 7| 100000| 100|0.38287010| PASSED
sts_serial| 7| 100000| 100|0.89902992| PASSED
sts_serial| 8| 100000| 100|0.76428076| PASSED
sts_serial| 8| 100000| 100|0.60644789| PASSED
sts_serial| 9| 100000| 100|0.48577631| PASSED
sts_serial| 9| 100000| 100|0.94892662| PASSED
sts_serial| 10| 100000| 100|0.61860146| PASSED
sts_serial| 10| 100000| 100|0.94390945| PASSED
sts_serial| 11| 100000| 100|0.91920037| PASSED
sts_serial| 11| 100000| 100|0.99395797| PASSED
sts_serial| 12| 100000| 100|0.53395979| PASSED
sts_serial| 12| 100000| 100|0.43408564| PASSED
sts_serial| 13| 100000| 100|0.97903729| PASSED
sts_serial| 13| 100000| 100|0.11918338| PASSED
sts_serial| 14| 100000| 100|0.57369729| PASSED
sts_serial| 14| 100000| 100|0.94441859| PASSED
sts_serial| 15| 100000| 100|0.74238375| PASSED
sts_serial| 15| 100000| 100|0.80171095| PASSED

```
          sts_serial| 16|    100000|     100|0.65597514| PASSED
          sts_serial| 16|    100000|     100|0.97650340| PASSED
         rgb_bitdist|  1|    100000|     100|0.29492087| PASSED
         rgb_bitdist|  2|    100000|     100|0.27700702| PASSED
         rgb_bitdist|  3|    100000|     100|0.85765540| PASSED
         rgb_bitdist|  4|    100000|     100|0.27348409| PASSED
         rgb_bitdist|  5|    100000|     100|0.89824772| PASSED
         rgb_bitdist|  6|    100000|     100|0.26415812| PASSED
         rgb_bitdist|  7|    100000|     100|0.77037999| PASSED
         rgb_bitdist|  8|    100000|     100|0.26742072| PASSED
         rgb_bitdist|  9|    100000|     100|0.88402132| PASSED
         rgb_bitdist| 10|    100000|     100|0.95113004| PASSED
         rgb_bitdist| 11|    100000|     100|0.99929189|   WEAK
         rgb_bitdist| 12|    100000|     100|0.30524379| PASSED
rgb_minimum_distance|  2|     10000|    1000|0.63626546| PASSED
rgb_minimum_distance|  3|     10000|    1000|0.03166243| PASSED
rgb_minimum_distance|  4|     10000|    1000|0.11175117| PASSED
rgb_minimum_distance|  5|     10000|    1000|0.54896237| PASSED
   rgb_permutations|  2|    100000|     100|0.54310799| PASSED
   rgb_permutations|  3|    100000|     100|0.21326680| PASSED
   rgb_permutations|  4|    100000|     100|0.48455959| PASSED
   rgb_permutations|  5|    100000|     100|0.41791317| PASSED
     rgb_lagged_sum|  0|   1000000|     100|0.69057006| PASSED
     rgb_lagged_sum|  1|   1000000|     100|0.34464436| PASSED
     rgb_lagged_sum|  2|   1000000|     100|0.69449348| PASSED
     rgb_lagged_sum|  3|   1000000|     100|0.12717354| PASSED
     rgb_lagged_sum|  4|   1000000|     100|0.74797773| PASSED
     rgb_lagged_sum|  5|   1000000|     100|0.81802493| PASSED
     rgb_lagged_sum|  6|   1000000|     100|0.71608758| PASSED
     rgb_lagged_sum|  7|   1000000|     100|0.66724403| PASSED
     rgb_lagged_sum|  8|   1000000|     100|0.23342893| PASSED
     rgb_lagged_sum|  9|   1000000|     100|0.00017500|   WEAK
     rgb_lagged_sum| 10|   1000000|     100|0.37401547| PASSED
     rgb_lagged_sum| 11|   1000000|     100|0.05048872| PASSED
     rgb_lagged_sum| 12|   1000000|     100|0.35113788| PASSED
     rgb_lagged_sum| 13|   1000000|     100|0.82844600| PASSED
     rgb_lagged_sum| 14|   1000000|     100|0.56205313| PASSED
     rgb_lagged_sum| 15|   1000000|     100|0.70647346| PASSED
     rgb_lagged_sum| 16|   1000000|     100|0.45957050| PASSED
     rgb_lagged_sum| 17|   1000000|     100|0.59232299| PASSED
     rgb_lagged_sum| 18|   1000000|     100|0.98040218| PASSED
     rgb_lagged_sum| 19|   1000000|     100|0.00122508|   WEAK
     rgb_lagged_sum| 20|   1000000|     100|0.03858470| PASSED
     rgb_lagged_sum| 21|   1000000|     100|0.23432881| PASSED
     rgb_lagged_sum| 22|   1000000|     100|0.72650681| PASSED
     rgb_lagged_sum| 23|   1000000|     100|0.18866686| PASSED
     rgb_lagged_sum| 24|   1000000|     100|0.00578472| PASSED
     rgb_lagged_sum| 25|   1000000|     100|0.25379262| PASSED
     rgb_lagged_sum| 26|   1000000|     100|0.76944138| PASSED
     rgb_lagged_sum| 27|   1000000|     100|0.35734523| PASSED
     rgb_lagged_sum| 28|   1000000|     100|0.65387832| PASSED
```

```
      rgb_lagged_sum| 29|   1000000|     100|0.00006339|   WEAK
      rgb_lagged_sum| 30|   1000000|     100|0.89407780| PASSED
      rgb_lagged_sum| 31|   1000000|     100|0.66679628| PASSED
      rgb_lagged_sum| 32|   1000000|     100|0.28033375| PASSED
     rgb_kstest_test|  0|     10000|    1000|0.14673439| PASSED
     dab_bytedistrib|  0| 51200000|       1|0.77961608| PASSED
             dab_dct| 256|     50000|       1|0.26794033| PASSED
Preparing to run test 207. ntuple = 0
        dab_filltree| 32| 15000000|       1|0.73912204| PASSED
        dab_filltree| 32| 15000000|       1|0.98689253| PASSED
Preparing to run test 208. ntuple = 0
       dab_filltree2|  0|  5000000|       1|0.72714106| PASSED
       dab_filltree2|  1|  5000000|       1|0.10741773| PASSED
Preparing to run test 209. ntuple = 0
        dab_monobit2| 12| 65000000|       1|0.77219213| PASSED
```

A8
Rng source: Espressif ESP32 hardware random
Test suite : DieHarder
Samples amount: 2000000000 octets
Results are below:

```
#=============================================================================
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================
   rng_name    |           filename             |rands/second|
file_input_raw|             ./esp32hrng-2Gb.dat| 2.87e+07 |
#=============================================================================
        test_name   |ntup| tsamples |psamples| p-value |Assessment
#=============================================================================
   diehard_birthdays|   0|       100|     100|0.79293668| PASSED
      diehard_operm5|   0|   1000000|     100|0.37355964| PASSED
  diehard_rank_32x32|   0|     40000|     100|0.70028394| PASSED
    diehard_rank_6x8|   0|    100000|     100|0.23999009| PASSED
   diehard_bitstream|   0|   2097152|     100|0.96641189| PASSED
        diehard_opso|   0|   2097152|     100|0.65341149| PASSED
        diehard_oqso|   0|   2097152|     100|0.79415168| PASSED
         diehard_dna|   0|   2097152|     100|0.07452452| PASSED
diehard_count_1s_str|   0|    256000|     100|0.29527737| PASSED
diehard_count_1s_byt|   0|    256000|     100|0.82027112| PASSED
diehard_parking_lot|   0|     12000|     100|0.03448968| PASSED
    diehard_2dsphere|   2|      8000|     100|0.35326843| PASSED
    diehard_3dsphere|   3|      4000|     100|0.79676503| PASSED
    diehard_squeeze|   0|    100000|     100|0.34316531| PASSED
        diehard_sums|   0|       100|     100|0.12449194| PASSED
        diehard_runs|   0|    100000|     100|0.44184899| PASSED
        diehard_runs|   0|    100000|     100|0.77480403| PASSED
       diehard_craps|   0|    200000|     100|0.93351042| PASSED
       diehard_craps|   0|    200000|     100|0.03957394| PASSED
marsaglia_tsang_gcd|   0| 10000000|     100|0.01763470| PASSED
marsaglia_tsang_gcd|   0| 10000000|     100|0.21523917| PASSED
```

sts_monobit	1	100000	100	0.27263890	PASSED
sts_runs	2	100000	100	0.48605330	PASSED
sts_serial	1	100000	100	0.52784304	PASSED
sts_serial	2	100000	100	0.04969853	PASSED
sts_serial	3	100000	100	0.45596449	PASSED
sts_serial	3	100000	100	0.99422527	PASSED
sts_serial	4	100000	100	0.88364507	PASSED
sts_serial	4	100000	100	0.04295833	PASSED
sts_serial	5	100000	100	0.48179792	PASSED
sts_serial	5	100000	100	0.53222696	PASSED
sts_serial	6	100000	100	0.60752302	PASSED
sts_serial	6	100000	100	0.86631836	PASSED
sts_serial	7	100000	100	0.17039001	PASSED
sts_serial	7	100000	100	0.65511120	PASSED
sts_serial	8	100000	100	0.26099188	PASSED
sts_serial	8	100000	100	0.77761536	PASSED
sts_serial	9	100000	100	0.95915415	PASSED
sts_serial	9	100000	100	0.27308521	PASSED
sts_serial	10	100000	100	0.71816475	PASSED
sts_serial	10	100000	100	0.77855619	PASSED
sts_serial	11	100000	100	0.31653281	PASSED
sts_serial	11	100000	100	0.99294396	PASSED
sts_serial	12	100000	100	0.12213496	PASSED
sts_serial	12	100000	100	0.34724821	PASSED
sts_serial	13	100000	100	0.24192076	PASSED
sts_serial	13	100000	100	0.94086151	PASSED
sts_serial	14	100000	100	0.45357846	PASSED
sts_serial	14	100000	100	0.45226277	PASSED
sts_serial	15	100000	100	0.36843621	PASSED
sts_serial	15	100000	100	0.96904358	PASSED
sts_serial	16	100000	100	0.07902768	PASSED
sts_serial	16	100000	100	0.23760700	PASSED
rgb_bitdist	1	100000	100	0.43864045	PASSED
rgb_bitdist	2	100000	100	0.99989235	WEAK
rgb_bitdist	3	100000	100	0.38313524	PASSED
rgb_bitdist	4	100000	100	0.84827358	PASSED
rgb_bitdist	5	100000	100	0.26703249	PASSED
rgb_bitdist	6	100000	100	0.78327235	PASSED
rgb_bitdist	7	100000	100	0.35789098	PASSED
rgb_bitdist	8	100000	100	0.69038573	PASSED
rgb_bitdist	9	100000	100	0.92211622	PASSED
rgb_bitdist	10	100000	100	0.23570793	PASSED
rgb_bitdist	11	100000	100	0.21784472	PASSED
rgb_bitdist	12	100000	100	0.26159606	PASSED
rgb_minimum_distance	2	10000	1000	0.75426035	PASSED
rgb_minimum_distance	3	10000	1000	0.07648435	PASSED
rgb_minimum_distance	4	10000	1000	0.92678140	PASSED
rgb_minimum_distance	5	10000	1000	0.10475742	PASSED
rgb_permutations	2	100000	100	0.77981037	PASSED
rgb_permutations	3	100000	100	0.28082501	PASSED
rgb_permutations	4	100000	100	0.87607884	PASSED

```
   rgb_permutations|   5|    100000|     100|0.85870232| PASSED
     rgb_lagged_sum|   0|   1000000|     100|0.42473894| PASSED
     rgb_lagged_sum|   1|   1000000|     100|0.82207681| PASSED
     rgb_lagged_sum|   2|   1000000|     100|0.17888475| PASSED
     rgb_lagged_sum|   3|   1000000|     100|0.94781462| PASSED
     rgb_lagged_sum|   4|   1000000|     100|0.37373831| PASSED
     rgb_lagged_sum|   5|   1000000|     100|0.24773264| PASSED
     rgb_lagged_sum|   6|   1000000|     100|0.98626340| PASSED
     rgb_lagged_sum|   7|   1000000|     100|0.62805908| PASSED
     rgb_lagged_sum|   8|   1000000|     100|0.99374971| PASSED
     rgb_lagged_sum|   9|   1000000|     100|0.11552562| PASSED
     rgb_lagged_sum|  10|   1000000|     100|0.98299815| PASSED
     rgb_lagged_sum|  11|   1000000|     100|0.52345530| PASSED
     rgb_lagged_sum|  12|   1000000|     100|0.48777072| PASSED
     rgb_lagged_sum|  13|   1000000|     100|0.31399091| PASSED
     rgb_lagged_sum|  14|   1000000|     100|0.79319313| PASSED
     rgb_lagged_sum|  15|   1000000|     100|0.53833885| PASSED
     rgb_lagged_sum|  16|   1000000|     100|0.24094507| PASSED
     rgb_lagged_sum|  17|   1000000|     100|0.23697953| PASSED
     rgb_lagged_sum|  18|   1000000|     100|0.25963022| PASSED
     rgb_lagged_sum|  19|   1000000|     100|0.00008640|   WEAK
     rgb_lagged_sum|  20|   1000000|     100|0.18330477| PASSED
     rgb_lagged_sum|  21|   1000000|     100|0.12604100| PASSED
     rgb_lagged_sum|  22|   1000000|     100|0.99854743|   WEAK
     rgb_lagged_sum|  23|   1000000|     100|0.90145641| PASSED
     rgb_lagged_sum|  24|   1000000|     100|0.24392568| PASSED
     rgb_lagged_sum|  25|   1000000|     100|0.18352561| PASSED
     rgb_lagged_sum|  26|   1000000|     100|0.19714039| PASSED
     rgb_lagged_sum|  27|   1000000|     100|0.96595665| PASSED
```

A9
Rng source: Russian “Sobol" hardware random generator
Test suite : DieHarder
Samples amount: 2000 000 000 octets
Results are below:

```
#=============================================================================
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================
   rng_name    |           filename             |rands/second|
file_input_raw|          ./sobol_plus_cpcsp.dat| 1.58e+07 |
#=============================================================================
        test_name   |ntup| tsamples |psamples| p-value |Assessment
#=============================================================================
   diehard_birthdays|   0|       100|     100|0.78204236| PASSED
      diehard_operm5|   0|   1000000|     100|0.45010721| PASSED
  diehard_rank_32x32|   0|     40000|     100|0.55849036| PASSED
    diehard_rank_6x8|   0|    100000|     100|0.45359454| PASSED
   diehard_bitstream|   0|   2097152|     100|0.81277604| PASSED
        diehard_opso|   0|   2097152|     100|0.75541881| PASSED
        diehard_oqso|   0|   2097152|     100|0.19690970| PASSED
         diehard_dna|   0|   2097152|     100|0.75967999| PASSED
```

diehard_count_1s_str	0	256000	100	0.83154348	PASSED
diehard_count_1s_byt	0	256000	100	0.78030040	PASSED
diehard_parking_lot	0	12000	100	0.32092307	PASSED
diehard_2dsphere	2	8000	100	0.98852941	PASSED
diehard_3dsphere	3	4000	100	0.80081617	PASSED
diehard_squeeze	0	100000	100	0.12481230	PASSED
diehard_sums	0	100	100	0.07094024	PASSED
diehard_runs	0	100000	100	0.44528526	PASSED
diehard_runs	0	100000	100	0.35602970	PASSED
diehard_craps	0	200000	100	0.41520000	PASSED
diehard_craps	0	200000	100	0.21332715	PASSED
marsaglia_tsang_gcd	0	10000000	100	0.09512726	PASSED
marsaglia_tsang_gcd	0	10000000	100	0.00003220	WEAK
sts_monobit	1	100000	100	0.38350807	PASSED
sts_runs	2	100000	100	0.00441727	WEAK
sts_serial	1	100000	100	0.84298490	PASSED
sts_serial	2	100000	100	0.14709854	PASSED
sts_serial	3	100000	100	0.05810013	PASSED
sts_serial	3	100000	100	0.17338307	PASSED
sts_serial	4	100000	100	0.03374129	PASSED
sts_serial	4	100000	100	0.58919552	PASSED
sts_serial	5	100000	100	0.61201922	PASSED
sts_serial	5	100000	100	0.30224012	PASSED
sts_serial	6	100000	100	0.96828265	PASSED
sts_serial	6	100000	100	0.03352300	PASSED
sts_serial	7	100000	100	0.99786343	WEAK
sts_serial	7	100000	100	0.38653769	PASSED
sts_serial	8	100000	100	0.51012687	PASSED
sts_serial	8	100000	100	0.36464155	PASSED
sts_serial	9	100000	100	0.22669143	PASSED
sts_serial	9	100000	100	0.26244525	PASSED
sts_serial	10	100000	100	0.68300598	PASSED
sts_serial	10	100000	100	0.25077033	PASSED
sts_serial	11	100000	100	0.69802932	PASSED
sts_serial	11	100000	100	0.40451864	PASSED
sts_serial	12	100000	100	0.75387808	PASSED
sts_serial	12	100000	100	0.28433614	PASSED
sts_serial	13	100000	100	0.93182652	PASSED
sts_serial	13	100000	100	0.92690068	PASSED
sts_serial	14	100000	100	0.94381019	PASSED
sts_serial	14	100000	100	0.57111510	PASSED
sts_serial	15	100000	100	0.39684834	PASSED
sts_serial	15	100000	100	0.10417342	PASSED
sts_serial	16	100000	100	0.99701546	WEAK
sts_serial	16	100000	100	0.44240482	PASSED
rgb_bitdist	1	100000	100	0.42611008	PASSED
rgb_bitdist	2	100000	100	0.28888301	PASSED
rgb_bitdist	3	100000	100	0.31123911	PASSED
rgb_bitdist	4	100000	100	0.23585413	PASSED
rgb_bitdist	5	100000	100	0.96036285	PASSED
rgb_bitdist	6	100000	100	0.82780774	PASSED

```
        rgb_bitdist|   7|    100000|     100|0.63243672| PASSED
        rgb_bitdist|   8|    100000|     100|0.72970680| PASSED
        rgb_bitdist|   9|    100000|     100|0.18697166| PASSED
        rgb_bitdist|  10|    100000|     100|0.37673220| PASSED
        rgb_bitdist|  11|    100000|     100|0.95159641| PASSED
        rgb_bitdist|  12|    100000|     100|0.81007185| PASSED
rgb_minimum_distance|   2|     10000|    1000|0.74074402| PASSED
rgb_minimum_distance|   3|     10000|    1000|0.94164160| PASSED
rgb_minimum_distance|   4|     10000|    1000|0.38632303| PASSED
rgb_minimum_distance|   5|     10000|    1000|0.12089324| PASSED
   rgb_permutations|   2|    100000|     100|0.04238411| PASSED
   rgb_permutations|   3|    100000|     100|0.91081139| PASSED
   rgb_permutations|   4|    100000|     100|0.83953741| PASSED
   rgb_permutations|   5|    100000|     100|0.49611347| PASSED
     rgb_lagged_sum|   0|   1000000|     100|0.90630836| PASSED
     rgb_lagged_sum|   1|   1000000|     100|0.76589734| PASSED
     rgb_lagged_sum|   2|   1000000|     100|0.13325210| PASSED
     rgb_lagged_sum|   3|   1000000|     100|0.60789333| PASSED
     rgb_lagged_sum|   4|   1000000|     100|0.99938102|   WEAK
     rgb_lagged_sum|   5|   1000000|     100|0.80057962| PASSED
     rgb_lagged_sum|   6|   1000000|     100|0.34529240| PASSED
     rgb_lagged_sum|   7|   1000000|     100|0.04996668| PASSED
     rgb_lagged_sum|   8|   1000000|     100|0.98885111| PASSED
     rgb_lagged_sum|   9|   1000000|     100|0.25450343| PASSED
     rgb_lagged_sum|  10|   1000000|     100|0.04694444| PASSED
     rgb_lagged_sum|  11|   1000000|     100|0.80549027| PASSED
     rgb_lagged_sum|  12|   1000000|     100|0.50855536| PASSED
     rgb_lagged_sum|  13|   1000000|     100|0.87024412| PASSED
     rgb_lagged_sum|  14|   1000000|     100|0.96525074| PASSED
     rgb_lagged_sum|  15|   1000000|     100|0.09905102| PASSED
     rgb_lagged_sum|  16|   1000000|     100|0.48389864| PASSED
     rgb_lagged_sum|  17|   1000000|     100|0.72076806| PASSED
     rgb_lagged_sum|  18|   1000000|     100|0.13911474| PASSED
     rgb_lagged_sum|  19|   1000000|     100|0.05886282| PASSED
     rgb_lagged_sum|  20|   1000000|     100|0.98047460| PASSED
     rgb_lagged_sum|  21|   1000000|     100|0.49959468| PASSED
     rgb_lagged_sum|  22|   1000000|     100|0.98657901| PASSED
     rgb_lagged_sum|  23|   1000000|     100|0.45414988| PASSED
     rgb_lagged_sum|  24|   1000000|     100|0.05836923| PASSED
     rgb_lagged_sum|  25|   1000000|     100|0.41070912| PASSED
     rgb_lagged_sum|  26|   1000000|     100|0.26152515| PASSED
     rgb_lagged_sum|  27|   1000000|     100|0.24165835| PASSED
     rgb_lagged_sum|  28|   1000000|     100|0.55164809| PASSED
     rgb_lagged_sum|  29|   1000000|     100|0.13428453| PASSED
     rgb_lagged_sum|  30|   1000000|     100|0.13120854| PASSED
     rgb_lagged_sum|  31|   1000000|     100|0.00045198|   WEAK
     rgb_lagged_sum|  32|   1000000|     100|0.19136228| PASSED
   rgb_kstest_test|   0|     10000|    1000|0.77223466| PASSED
   dab_bytedistrib|   0| 51200000|       1|0.48388855| PASSED
           dab_dct| 256|     50000|       1|0.90210642| PASSED
Preparing to run test 207. ntuple = 0
```

```
      dab_filltree| 32| 15000000|       1|0.76706006| PASSED
      dab_filltree| 32| 15000000|       1|0.65117512| PASSED
Preparing to run test 208. ntuple = 0
     dab_filltree2|  0|  5000000|       1|0.12149536| PASSED
     dab_filltree2|  1|  5000000|       1|0.45756303| PASSED
Preparing to run test 209. ntuple = 0
      dab_monobit2| 12| 65000000|       1|0.98462619| PASSED
```

A10
Rng source: Russian “Sobol" hardware random generator
Test suite : NIST
Samples amount: 120000000 octets
Results are below:

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

---

generator is </home/peter/projects/data/sobol_plus_cpcsp.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
3	0	2	1	0	1	1	1	1	0	0.534146	10/10	Frequency
1	2	0	0	1	1	1	0	1	3	0.534146	10/10	BlockFrequency
2	2	0	0	2	1	0	1	2	0	0.534146	10/10	CumulativeSums
3	1	3	0	1	0	1	1	0	0	0.213309	10/10	CumulativeSums
0	2	2	0	2	0	1	2	0	1	0.534146	10/10	Runs
0	0	1	3	1	0	2	3	0	0	0.122325	10/10	LongestRun
1	0	0	0	2	1	0	2	3	1	0.350485	10/10	Rank
0	0	3	0	0	3	1	2	0	1	0.122325	10/10	FFT
1	1	1	0	0	0	2	0	2	3	0.350485	10/10	NonOverlappingTemplate
0	0	0	1	3	2	1	1	0	2	0.350485	10/10	NonOverlappingTemplate
1	1	1	1	1	0	1	3	0	1	0.739918	9/10	NonOverlappingTemplate
2	0	0	2	0	0	1	2	0	3	0.213309	10/10	NonOverlappingTemplate
0	0	2	2	1	1	0	0	1	3	0.350485	10/10	NonOverlappingTemplate
1	1	2	1	0	1	3	0	1	0	0.534146	10/10	NonOverlappingTemplate
0	0	2	3	0	2	0	1	0	2	0.213309	10/10	NonOverlappingTemplate
2	0	2	2	3	0	0	0	1	0	0.213309	9/10	NonOverlappingTemplate
0	3	0	1	1	1	1	2	0	1	0.534146	10/10	NonOverlappingTemplate
1	0	1	0	2	0	0	2	2	2	0.534146	10/10	NonOverlappingTemplate
2	1	0	1	1	1	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
1	1	1	2	0	0	1	2	1	1	0.911413	10/10	NonOverlappingTemplate
0	3	0	1	0	3	1	0	0	2	0.122325	10/10	NonOverlappingTemplate
2	1	0	2	0	1	0	2	1	1	0.739918	10/10	NonOverlappingTemplate
1	2	1	0	1	1	0	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	3	1	1	2	1	1	0	0	0	0.534146	10/10	NonOverlappingTemplate
2	0	1	0	0	1	2	2	1	1	0.739918	10/10	NonOverlappingTemplate
1	2	1	1	2	1	1	0	1	0	0.911413	10/10	NonOverlappingTemplate
2	0	2	0	0	1	1	0	3	1	0.350485	9/10	NonOverlappingTemplate
0	0	2	1	3	1	1	1	0	1	0.534146	10/10	NonOverlappingTemplate

2	1	0	1	2	1	1	0	1	1	0.911413	10/10	NonOverlappingTemplate
3	0	1	2	0	1	1	0	1	1	0.534146	10/10	NonOverlappingTemplate
3	1	1	1	0	2	1	0	1	0	0.534146	10/10	NonOverlappingTemplate
0	2	0	1	0	2	1	3	1	0	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	0	2	2	0	1	2	0.739918	10/10	NonOverlappingTemplate
1	1	0	3	0	1	2	0	1	1	0.534146	10/10	NonOverlappingTemplate
1	2	2	2	0	0	2	0	0	1	0.534146	10/10	NonOverlappingTemplate
1	2	2	0	0	1	0	2	2	0	0.534146	10/10	NonOverlappingTemplate
0	0	0	4	1	1	1	3	0	0	0.035174	10/10	NonOverlappingTemplate
0	0	3	0	2	0	1	2	0	2	0.213309	10/10	NonOverlappingTemplate
1	2	3	0	0	0	3	0	1	0	0.122325	10/10	NonOverlappingTemplate
1	0	0	1	0	2	1	1	2	2	0.739918	10/10	NonOverlappingTemplate
2	1	0	0	2	1	2	0	0	2	0.534146	10/10	NonOverlappingTemplate
0	1	4	0	3	0	0	0	1	1	0.035174	10/10	NonOverlappingTemplate
1	2	1	0	2	2	0	1	1	0	0.739918	10/10	NonOverlappingTemplate
0	0	0	2	1	1	1	1	2	2	0.739918	10/10	NonOverlappingTemplate
1	2	1	4	0	1	1	0	0	0	0.122325	10/10	NonOverlappingTemplate
1	3	0	0	1	0	1	0	1	3	0.213309	10/10	NonOverlappingTemplate
0	1	0	1	0	2	1	2	1	2	0.739918	10/10	NonOverlappingTemplate
1	2	0	3	2	1	0	1	0	0	0.350485	10/10	NonOverlappingTemplate
3	1	0	1	0	2	0	1	1	1	0.534146	10/10	NonOverlappingTemplate
0	2	2	2	1	2	0	0	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	3	0	0	1	1	1	2	0	0.534146	10/10	NonOverlappingTemplate
0	1	0	1	1	1	2	3	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	0	2	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
3	1	1	0	1	3	0	0	1	0	0.213309	10/10	NonOverlappingTemplate
3	1	0	1	2	0	2	0	0	1	0.350485	10/10	NonOverlappingTemplate
1	2	0	1	1	1	1	0	2	1	0.911413	10/10	NonOverlappingTemplate
1	0	3	0	1	1	0	2	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	0	2	6	0	0	0	0	1	0.000199	10/10	NonOverlappingTemplate
1	0	0	1	0	0	2	0	4	2	0.066882	10/10	NonOverlappingTemplate
1	2	0	1	1	1	1	1	0	2	0.911413	10/10	NonOverlappingTemplate
2	2	0	1	0	0	2	0	2	1	0.534146	10/10	NonOverlappingTemplate
2	1	0	3	0	0	1	0	3	0	0.122325	10/10	NonOverlappingTemplate
1	1	0	1	1	0	0	1	2	3	0.534146	10/10	NonOverlappingTemplate
0	1	2	2	1	0	2	0	0	2	0.534146	10/10	NonOverlappingTemplate
1	1	0	1	1	1	1	0	1	3	0.739918	10/10	NonOverlappingTemplate
0	3	0	0	2	1	1	1	2	0	0.350485	10/10	NonOverlappingTemplate
2	0	0	1	1	2	2	0	0	2	0.534146	10/10	NonOverlappingTemplate
0	1	1	1	0	1	2	1	1	2	0.911413	10/10	NonOverlappingTemplate
0	0	1	1	1	1	2	2	2	0	0.739918	10/10	NonOverlappingTemplate
0	0	3	0	2	0	1	1	1	2	0.350485	10/10	NonOverlappingTemplate
1	2	1	0	1	2	2	1	0	0	0.739918	10/10	NonOverlappingTemplate
0	2	2	1	1	0	2	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	1	2	1	0	3	1	0	1	1	0.534146	10/10	NonOverlappingTemplate
0	0	2	2	1	0	1	2	2	0	0.534146	10/10	NonOverlappingTemplate
2	0	0	1	2	1	3	0	1	0	0.350485	10/10	NonOverlappingTemplate
2	1	1	2	1	2	0	1	0	0	0.739918	10/10	NonOverlappingTemplate
2	2	1	2	1	0	0	1	1	0	0.739918	10/10	NonOverlappingTemplate
0	1	2	3	0	1	0	2	0	1	0.350485	10/10	NonOverlappingTemplate
2	0	0	1	1	1	2	1	2	0	0.739918	10/10	NonOverlappingTemplate

0	2	1	0	1	1	1	2	2	0	0.739918	10/10	NonOverlappingTemplate
0	1	3	2	0	1	0	2	0	1	0.350485	10/10	NonOverlappingTemplate
3	1	2	1	0	0	2	1	0	0	0.350485	10/10	NonOverlappingTemplate
1	1	1	0	0	0	2	0	2	3	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	1	1	1	2	1	2	0.911413	10/10	NonOverlappingTemplate
2	2	0	1	0	0	2	0	2	1	0.534146	9/10	NonOverlappingTemplate
2	0	1	0	2	2	0	2	0	1	0.534146	10/10	NonOverlappingTemplate
1	3	2	1	0	0	0	1	1	1	0.534146	10/10	NonOverlappingTemplate
0	2	1	1	2	0	0	1	3	0	0.350485	10/10	NonOverlappingTemplate
1	0	2	1	1	0	2	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	1	1	1	1	2	3	1	0	0.534146	10/10	NonOverlappingTemplate
0	0	3	1	2	1	0	1	2	0	0.350485	10/10	NonOverlappingTemplate
1	0	1	3	2	1	0	1	1	0	0.534146	10/10	NonOverlappingTemplate
0	0	2	0	2	1	1	2	1	1	0.739918	10/10	NonOverlappingTemplate
1	2	0	0	0	0	3	3	1	0	0.122325	9/10	NonOverlappingTemplate
1	4	2	0	0	1	2	0	0	0	0.066882	10/10	NonOverlappingTemplate
1	1	2	1	0	0	1	2	0	2	0.739918	10/10	NonOverlappingTemplate
1	2	1	1	0	2	1	1	0	1	0.911413	10/10	NonOverlappingTemplate
0	2	1	0	2	2	1	1	1	0	0.739918	10/10	NonOverlappingTemplate
1	0	2	0	2	0	3	0	1	1	0.350485	10/10	NonOverlappingTemplate
2	3	0	1	1	0	0	0	2	1	0.350485	10/10	NonOverlappingTemplate
0	1	2	1	0	3	0	1	2	0	0.350485	10/10	NonOverlappingTemplate
1	0	2	2	1	2	0	1	1	0	0.739918	10/10	NonOverlappingTemplate
1	0	3	0	2	1	0	0	1	2	0.350485	10/10	NonOverlappingTemplate
0	2	1	0	0	2	1	3	1	0	0.350485	10/10	NonOverlappingTemplate
0	0	1	2	0	1	2	0	1	3	0.350485	10/10	NonOverlappingTemplate
2	2	0	0	0	1	1	2	2	0	0.534146	10/10	NonOverlappingTemplate
2	1	1	1	1	2	0	0	0	2	0.739918	10/10	NonOverlappingTemplate
1	1	0	0	3	0	1	2	2	0	0.350485	10/10	NonOverlappingTemplate
2	1	1	0	2	0	0	1	2	1	0.739918	9/10	NonOverlappingTemplate
2	2	0	1	1	1	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
0	1	0	0	1	1	2	0	4	1	0.122325	10/10	NonOverlappingTemplate
0	4	1	0	0	1	0	1	2	1	0.122325	10/10	NonOverlappingTemplate
0	1	1	2	0	1	1	2	0	2	0.739918	10/10	NonOverlappingTemplate
2	0	2	2	3	0	1	0	0	0	0.213309	10/10	NonOverlappingTemplate
2	1	0	1	1	0	1	1	3	0	0.534146	10/10	NonOverlappingTemplate
0	3	1	1	1	0	0	2	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	0	1	3	2	2	0	0	1	0.350485	10/10	NonOverlappingTemplate
0	1	0	0	0	1	5	1	1	1	0.017912	10/10	NonOverlappingTemplate
3	1	0	2	1	2	0	0	0	1	0.350485	10/10	NonOverlappingTemplate
1	2	0	2	0	4	0	1	0	0	0.066882	10/10	NonOverlappingTemplate
1	0	1	1	0	1	0	2	3	1	0.534146	10/10	NonOverlappingTemplate
0	1	1	1	4	1	0	0	2	0	0.122325	10/10	NonOverlappingTemplate
0	4	0	0	0	3	1	0	1	1	0.035174	10/10	NonOverlappingTemplate
1	2	0	0	0	1	3	0	2	1	0.350485	10/10	NonOverlappingTemplate
2	0	0	0	1	3	1	1	2	0	0.350485	10/10	NonOverlappingTemplate
2	0	0	0	1	0	2	1	2	2	0.534146	10/10	NonOverlappingTemplate
0	3	3	0	1	2	0	0	0	1	0.122325	10/10	NonOverlappingTemplate
3	2	0	0	0	2	0	2	0	1	0.213309	10/10	NonOverlappingTemplate
2	0	2	1	2	0	0	2	1	0	0.534146	10/10	NonOverlappingTemplate
2	1	2	0	0	1	4	0	0	0	0.066882	9/10	NonOverlappingTemplate

0	0	5	0	0	0	1	1	3	0	0.002043	10/10	NonOverlappingTemplate
0	0	1	0	2	5	0	0	1	1	0.008879	10/10	NonOverlappingTemplate
0	2	0	1	1	2	1	1	0	2	0.739918	10/10	NonOverlappingTemplate
0	0	2	0	2	3	0	1	1	1	0.350485	10/10	NonOverlappingTemplate
2	2	1	0	0	0	2	0	2	1	0.534146	10/10	NonOverlappingTemplate
1	0	3	3	2	0	0	0	0	1	0.122325	10/10	NonOverlappingTemplate
3	2	0	0	0	1	1	1	0	2	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	0	1	1	1	1	4	0.213309	10/10	NonOverlappingTemplate
0	2	0	0	1	1	1	0	4	1	0.122325	10/10	NonOverlappingTemplate
1	0	1	2	1	2	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	0	2	2	1	1	1	0	1	1	0.911413	10/10	NonOverlappingTemplate
1	2	2	0	2	2	1	0	0	0	0.534146	10/10	NonOverlappingTemplate
1	0	2	0	1	1	0	1	3	1	0.534146	10/10	NonOverlappingTemplate
3	1	1	1	1	2	1	0	0	0	0.534146	10/10	NonOverlappingTemplate
1	0	1	2	0	1	3	0	2	0	0.350485	10/10	NonOverlappingTemplate
0	1	0	0	0	2	3	1	1	2	0.350485	10/10	NonOverlappingTemplate
2	1	0	0	2	0	1	1	1	2	0.739918	10/10	NonOverlappingTemplate
0	3	0	3	1	0	0	2	1	0	0.122325	10/10	NonOverlappingTemplate
0	3	0	1	1	2	0	1	0	2	0.350485	10/10	NonOverlappingTemplate
1	1	1	1	0	0	1	1	2	2	0.911413	10/10	NonOverlappingTemplate
1	1	0	3	2	1	0	0	1	1	0.534146	10/10	NonOverlappingTemplate
2	2	0	1	2	1	0	2	0	0	0.534146	9/10	NonOverlappingTemplate
0	2	2	2	1	2	1	0	0	0	0.534146	10/10	NonOverlappingTemplate
1	0	0	1	0	1	1	2	0	4	0.122325	10/10	NonOverlappingTemplate
1	4	2	2	0	1	0	0	0	0	0.066882	9/10	NonOverlappingTemplate
3	1	2	1	0	0	2	1	0	0	0.350485	10/10	NonOverlappingTemplate
6	2	0	0	0	1	1	0	0	0	0.000199	7/10	* OverlappingTemplate
1	1	2	1	0	1	0	0	3	1	0.534146	10/10	Universal
2	1	2	0	1	1	2	1	0	0	0.739918	10/10	ApproximateEntropy
3	1	1	1	0	1	1	1	1	0	0.739918	10/10	RandomExcursions
0	0	3	0	2	2	2	0	1	0	0.213309	10/10	RandomExcursions
0	2	4	0	0	3	0	0	0	1	0.017912	10/10	RandomExcursions
3	1	2	0	1	0	2	0	1	0	0.350485	10/10	RandomExcursions
4	1	0	1	1	1	0	0	2	0	0.122325	10/10	RandomExcursions
1	1	1	1	0	3	1	0	1	1	0.739918	9/10	RandomExcursions
1	3	1	0	1	2	0	0	2	0	0.350485	9/10	RandomExcursions
2	0	1	2	0	2	0	0	1	2	0.534146	9/10	RandomExcursions
1	1	0	0	1	5	1	0	0	1	0.017912	10/10	RandomExcursionsVariant
1	1	1	1	2	1	2	0	0	1	0.911413	10/10	RandomExcursionsVariant
1	0	1	2	1	2	0	0	1	2	0.739918	10/10	RandomExcursionsVariant
0	1	1	0	1	2	1	2	0	2	0.739918	10/10	RandomExcursionsVariant
1	0	0	1	1	2	1	1	1	2	0.911413	10/10	RandomExcursionsVariant
1	0	1	0	0	1	1	3	2	1	0.534146	10/10	RandomExcursionsVariant
1	0	2	1	0	0	2	2	1	1	0.739918	10/10	RandomExcursionsVariant
1	2	1	1	0	0	2	0	1	2	0.739918	10/10	RandomExcursionsVariant
2	0	0	0	1	5	0	1	1	0	0.008879	9/10	RandomExcursionsVariant
2	0	2	0	2	2	2	0	0	0	0.350485	10/10	RandomExcursionsVariant
2	1	0	2	1	3	1	0	0	0	0.350485	10/10	RandomExcursionsVariant
1	1	1	1	2	1	2	1	0	0	0.911413	10/10	RandomExcursionsVariant
1	1	1	0	1	4	0	0	2	0	0.122325	10/10	RandomExcursionsVariant
1	1	1	0	1	0	1	2	0	3	0.534146	9/10	RandomExcursionsVariant

1	0	1	0	1	1	1	1	3	1	0.739918	9/10	RandomExcursionsVariant
1	1	0	0	0	5	0	1	2	0	0.008879	9/10	RandomExcursionsVariant
2	0	0	0	3	0	2	1	1	1	0.350485	9/10	RandomExcursionsVariant
2	0	0	1	1	1	1	1	1	2	0.911413	9/10	RandomExcursionsVariant
0	2	0	2	1	4	0	0	0	1	0.066882	10/10	Serial
0	0	0	2	3	2	2	0	0	1	0.213309	10/10	Serial
2	0	0	0	1	0	2	3	1	1	0.350485	10/10	LinearComplexity

- -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 8 for a
sample size = 10 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 8 for a sample size = 10 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- -

A11
Rng source: Russian “Sobol" hardware random generator
Test suite : Ent
Samples amount: 120000000 octets
Results are below:

Entropy = 7.999999 bits per byte.

Optimum compression would reduce the size
of this 120000000 byte file by 0 percent.

Chi square distribution for 120000000 samples is 238.67, and randomly
would exceed this value 76.09 percent of the times.

Arithmetic mean value of data bytes is 127.4959 (127.5 = random).
Monte Carlo value for Pi is 3.141263000 (error 0.01 percent).
Serial correlation coefficient is -0.000057 (totally uncorrelated = 0.0).

A12
Rng source: ESP32-based hardware random source
Test suite : Ent
Samples amount: 120000000 octets
Results are below:

Entropy = 7.999998 bits per byte.

Optimum compression would reduce the size

of this 120000000 byte file by 0 percent.

Chi square distribution for 120000000 samples is 281.30, and randomly
would exceed this value 12.38 percent of the times.

Arithmetic mean value of data bytes is 127.5106 (127.5 = random).
Monte Carlo value for Pi is 3.140397200 (error 0.04 percent).
Serial correlation coefficient is -0.000013 (totally uncorrelated = 0.0).

A13

Rng source: Linux /dev/random
Test suite : NIST
Samples amount: 125829120 octets
Results are below:

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

---

generator is </home/peter/projects/data/dev_random.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
0	0	4	0	1	1	0	1	2	1	0.122325	10/10	Frequency
1	0	2	2	0	1	2	1	1	0	0.739918	10/10	BlockFrequency
0	0	1	2	4	1	0	0	0	2	0.066882	10/10	CumulativeSums
0	1	1	3	1	0	1	0	1	2	0.534146	10/10	CumulativeSums
3	2	1	0	0	2	0	1	0	1	0.350485	9/10	Runs
2	1	2	1	0	2	1	1	0	0	0.739918	10/10	LongestRun
1	1	3	1	2	1	0	1	0	0	0.534146	10/10	Rank
2	1	1	1	1	2	0	1	0	1	0.911413	9/10	FFT
1	1	0	1	1	2	0	2	1	1	0.911413	10/10	NonOverlappingTemplate
1	3	0	1	0	1	1	0	1	2	0.534146	10/10	NonOverlappingTemplate
1	0	1	1	0	0	1	2	2	2	0.739918	10/10	NonOverlappingTemplate
2	0	1	2	0	3	0	1	1	0	0.350485	10/10	NonOverlappingTemplate
1	2	1	0	1	1	1	0	2	1	0.911413	10/10	NonOverlappingTemplate
1	1	0	2	0	2	2	1	0	1	0.739918	9/10	NonOverlappingTemplate
0	1	0	2	1	2	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
0	0	3	0	0	1	2	1	1	2	0.350485	10/10	NonOverlappingTemplate
1	2	0	0	2	1	1	3	0	0	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	1	3	2	0	0	2	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	3	1	2	1	0	0	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	2	1	1	0	2	0	0.911413	10/10	NonOverlappingTemplate
0	0	0	1	0	2	3	1	2	1	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	1	1	3	1	1	0	0.739918	9/10	NonOverlappingTemplate
0	1	2	1	0	0	0	0	4	2	0.066882	10/10	NonOverlappingTemplate
1	1	2	2	0	0	1	1	0	2	0.739918	10/10	NonOverlappingTemplate
0	2	0	2	1	0	2	0	2	1	0.534146	10/10	NonOverlappingTemplate
1	1	0	1	2	1	0	2	1	1	0.911413	10/10	NonOverlappingTemplate
0	2	0	2	1	2	1	1	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	3	1	2	2	0	2	0	0	0.213309	10/10	NonOverlappingTemplate

1	3	1	1	0	1	1	1	1	0	0.739918	10/10	NonOverlappingTemplate
0	2	1	0	3	1	2	1	0	0	0.350485	10/10	NonOverlappingTemplate
1	0	1	3	0	1	1	1	2	0	0.534146	10/10	NonOverlappingTemplate
1	1	2	2	0	1	0	1	0	2	0.739918	10/10	NonOverlappingTemplate
0	0	0	3	1	2	2	1	1	0	0.350485	10/10	NonOverlappingTemplate
2	0	2	0	1	3	1	0	1	0	0.350485	10/10	NonOverlappingTemplate
1	1	1	0	1	2	1	1	2	0	0.911413	10/10	NonOverlappingTemplate
2	2	1	0	0	1	1	2	1	0	0.739918	10/10	NonOverlappingTemplate
1	0	1	2	2	2	1	0	1	0	0.739918	10/10	NonOverlappingTemplate
0	2	1	1	2	0	0	1	1	2	0.739918	10/10	NonOverlappingTemplate
2	0	1	1	1	0	1	0	2	2	0.739918	9/10	NonOverlappingTemplate
0	1	1	3	0	1	1	2	0	1	0.534146	10/10	NonOverlappingTemplate
1	3	1	0	0	2	1	1	1	0	0.534146	10/10	NonOverlappingTemplate
1	2	1	1	0	2	3	0	0	0	0.350485	10/10	NonOverlappingTemplate
0	1	1	1	1	0	1	1	2	2	0.911413	10/10	NonOverlappingTemplate
0	0	0	2	0	1	1	0	5	1	0.008879	10/10	NonOverlappingTemplate
0	0	0	4	1	0	1	2	2	0	0.066882	10/10	NonOverlappingTemplate
0	0	0	2	0	0	3	1	1	3	0.122325	10/10	NonOverlappingTemplate
2	1	3	1	0	0	1	1	0	1	0.534146	10/10	NonOverlappingTemplate
2	1	1	0	0	1	1	1	3	0	0.534146	10/10	NonOverlappingTemplate
2	3	0	1	0	0	2	0	2	0	0.213309	10/10	NonOverlappingTemplate
0	0	2	0	1	3	1	3	0	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	1	1	0	2	1	1	3	0.534146	10/10	NonOverlappingTemplate
1	1	1	0	1	1	1	1	0	3	0.739918	10/10	NonOverlappingTemplate
0	1	1	0	2	1	1	1	1	2	0.911413	10/10	NonOverlappingTemplate
0	2	1	1	1	2	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
0	1	0	0	1	1	2	3	1	1	0.534146	10/10	NonOverlappingTemplate
1	2	3	1	1	0	1	0	1	0	0.534146	10/10	NonOverlappingTemplate
4	0	3	1	0	1	0	1	0	0	0.035174	10/10	NonOverlappingTemplate
1	2	0	0	1	0	2	1	2	1	0.739918	10/10	NonOverlappingTemplate
1	0	2	0	1	0	4	0	0	2	0.066882	10/10	NonOverlappingTemplate
1	0	1	0	1	2	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
2	1	0	2	2	0	0	0	1	2	0.534146	9/10	NonOverlappingTemplate
0	0	2	2	1	3	0	1	1	0	0.350485	10/10	NonOverlappingTemplate
3	0	1	2	0	2	1	0	1	0	0.350485	10/10	NonOverlappingTemplate
0	4	4	0	0	0	1	1	0	0	0.004301	10/10	NonOverlappingTemplate
0	0	0	2	1	3	1	1	1	1	0.534146	10/10	NonOverlappingTemplate
2	2	0	0	1	0	1	1	1	2	0.739918	10/10	NonOverlappingTemplate
0	1	1	2	1	0	1	2	1	1	0.911413	10/10	NonOverlappingTemplate
0	2	2	1	1	0	1	2	1	0	0.739918	10/10	NonOverlappingTemplate
0	0	0	1	2	0	3	0	2	2	0.213309	10/10	NonOverlappingTemplate
1	3	0	2	0	1	0	2	1	0	0.350485	10/10	NonOverlappingTemplate
0	0	0	3	3	0	1	0	3	0	0.035174	10/10	NonOverlappingTemplate
1	2	1	0	2	0	0	2	2	0	0.534146	10/10	NonOverlappingTemplate
3	0	0	0	0	0	1	2	3	1	0.122325	10/10	NonOverlappingTemplate
2	3	0	1	1	0	0	0	0	3	0.122325	10/10	NonOverlappingTemplate
2	1	0	3	1	1	1	1	0	0	0.534146	10/10	NonOverlappingTemplate
0	0	2	2	0	3	2	1	0	0	0.213309	10/10	NonOverlappingTemplate
0	0	1	3	0	1	0	1	3	1	0.213309	10/10	NonOverlappingTemplate
0	0	3	0	2	0	4	1	0	0	0.017912	10/10	NonOverlappingTemplate
1	0	2	1	1	0	3	1	0	1	0.534146	10/10	NonOverlappingTemplate

2	1	2	1	0	0	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
0	1	1	1	0	2	4	0	1	0	0.122325	10/10	NonOverlappingTemplate
0	1	2	1	2	0	0	0	1	3	0.350485	10/10	NonOverlappingTemplate
1	1	0	1	1	2	0	2	1	1	0.911413	10/10	NonOverlappingTemplate
4	0	1	0	2	0	0	0	3	0	0.017912	10/10	NonOverlappingTemplate
2	0	2	0	1	1	2	0	0	2	0.534146	10/10	NonOverlappingTemplate
2	1	0	3	0	1	0	3	0	0	0.122325	10/10	NonOverlappingTemplate
0	0	3	1	1	1	1	1	2	0	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	0	1	0	1	0	4	0.213309	10/10	NonOverlappingTemplate
1	1	2	0	1	1	0	2	1	1	0.911413	10/10	NonOverlappingTemplate
2	4	1	1	1	0	1	0	0	0	0.122325	10/10	NonOverlappingTemplate
2	1	2	1	0	0	2	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	1	0	4	0	0	2	0	2	1	0.066882	10/10	NonOverlappingTemplate
0	3	1	2	1	2	0	1	0	0	0.350485	10/10	NonOverlappingTemplate
2	2	0	1	0	4	0	0	0	1	0.066882	10/10	NonOverlappingTemplate
2	1	1	1	2	0	1	0	1	1	0.911413	9/10	NonOverlappingTemplate
2	1	0	0	1	1	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
0	1	1	0	2	0	3	1	1	1	0.534146	10/10	NonOverlappingTemplate
1	2	0	2	1	2	0	0	0	2	0.534146	10/10	NonOverlappingTemplate
2	0	1	0	1	1	2	0	1	2	0.739918	10/10	NonOverlappingTemplate
2	1	0	0	1	3	0	1	1	1	0.534146	10/10	NonOverlappingTemplate
0	2	1	2	1	0	0	2	2	0	0.534146	10/10	NonOverlappingTemplate
3	2	1	1	2	0	0	0	0	1	0.350485	10/10	NonOverlappingTemplate
0	0	0	1	1	1	0	3	3	1	0.213309	10/10	NonOverlappingTemplate
0	1	1	3	0	2	0	0	2	1	0.350485	10/10	NonOverlappingTemplate
1	1	1	0	1	1	3	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	0	0	1	2	1	0	2	3	1	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	0	1	2	2	0	2	0.739918	10/10	NonOverlappingTemplate
2	1	1	1	1	1	2	0	0	1	0.911413	10/10	NonOverlappingTemplate
0	4	0	3	0	0	0	1	2	0	0.017912	10/10	NonOverlappingTemplate
1	1	0	1	0	2	2	1	2	0	0.739918	10/10	NonOverlappingTemplate
0	0	2	0	2	1	0	2	2	1	0.534146	10/10	NonOverlappingTemplate
0	3	2	0	2	2	0	1	0	0	0.213309	10/10	NonOverlappingTemplate
2	1	1	0	1	0	2	0	2	1	0.739918	10/10	NonOverlappingTemplate
1	1	0	1	0	1	2	1	1	2	0.911413	10/10	NonOverlappingTemplate
1	1	2	1	2	0	0	0	2	1	0.739918	10/10	NonOverlappingTemplate
1	0	3	0	5	0	0	1	0	0	0.002043	9/10	NonOverlappingTemplate
1	1	1	1	0	1	0	1	3	1	0.739918	10/10	NonOverlappingTemplate
0	2	1	0	1	0	1	3	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	1	4	0	1	0	2	1	0	0.122325	9/10	NonOverlappingTemplate
0	1	1	1	0	0	1	1	3	2	0.534146	10/10	NonOverlappingTemplate
1	0	1	1	2	0	2	1	2	0	0.739918	10/10	NonOverlappingTemplate
1	0	1	1	0	2	1	1	1	2	0.911413	9/10	NonOverlappingTemplate
1	0	1	2	0	2	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	2	2	1	0	0	1	1	0.911413	10/10	NonOverlappingTemplate
2	1	1	2	0	1	0	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	2	1	0	0	3	1	3	0	0.122325	10/10	NonOverlappingTemplate
1	1	1	0	2	1	1	2	0	1	0.911413	10/10	NonOverlappingTemplate
1	1	1	0	3	0	0	1	2	1	0.534146	10/10	NonOverlappingTemplate
1	2	1	2	0	1	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
1	2	1	1	0	3	0	0	2	0	0.350485	9/10	NonOverlappingTemplate

1	1	1	1	0	2	2	0	2	0	0.739918	10/10	NonOverlappingTemplate
1	1	0	1	0	1	3	1	1	1	0.739918	10/10	NonOverlappingTemplate
1	0	3	0	0	0	2	4	0	0	0.017912	10/10	NonOverlappingTemplate
2	0	2	1	1	0	0	1	2	1	0.739918	10/10	NonOverlappingTemplate
3	1	1	1	1	0	0	0	2	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	0	2	2	0	3	1	1	0.350485	10/10	NonOverlappingTemplate
1	0	0	4	1	0	1	0	1	2	0.122325	10/10	NonOverlappingTemplate
1	2	1	1	1	1	2	0	1	0	0.911413	10/10	NonOverlappingTemplate
1	2	1	0	0	1	0	1	4	0	0.122325	9/10	NonOverlappingTemplate
1	0	1	1	1	3	0	2	1	0	0.534146	10/10	NonOverlappingTemplate
2	0	2	0	2	0	0	2	1	1	0.534146	10/10	NonOverlappingTemplate
0	0	1	0	4	2	1	2	0	0	0.066882	10/10	NonOverlappingTemplate
1	1	0	3	1	0	2	0	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	3	2	0	0	2	1	1	0.350485	10/10	NonOverlappingTemplate
3	0	0	3	0	0	1	3	0	0	0.035174	9/10	NonOverlappingTemplate
0	0	2	1	1	1	2	1	0	2	0.739918	10/10	NonOverlappingTemplate
0	0	3	1	1	0	0	1	2	2	0.350485	10/10	NonOverlappingTemplate
0	2	2	0	0	1	1	2	0	2	0.534146	10/10	NonOverlappingTemplate
0	1	2	1	1	1	3	0	1	0	0.534146	10/10	NonOverlappingTemplate
2	1	1	0	0	2	2	0	0	2	0.534146	9/10	NonOverlappingTemplate
3	0	0	1	0	1	0	2	2	1	0.350485	10/10	NonOverlappingTemplate
1	0	0	0	1	1	2	0	4	1	0.122325	10/10	NonOverlappingTemplate
0	4	0	0	0	2	1	1	2	0	0.066882	10/10	NonOverlappingTemplate
1	2	1	0	0	1	2	0	1	2	0.739918	10/10	NonOverlappingTemplate
0	1	0	2	1	1	3	1	0	1	0.534146	10/10	NonOverlappingTemplate
0	1	2	1	2	0	0	0	1	3	0.350485	10/10	NonOverlappingTemplate
8	1	0	0	1	0	0	0	0	0	0.000000 *	9/10	OverlappingTemplate
0	3	0	1	0	2	1	1	1	1	0.534146	10/10	Universal
1	3	1	2	0	1	0	0	0	2	0.350485	10/10	ApproximateEntropy
1	1	0	0	1	1	3	1	1	1	0.739918	10/10	RandomExcursions
0	2	0	1	0	2	1	1	2	1	0.739918	10/10	RandomExcursions
1	2	1	3	0	0	1	1	1	0	0.534146	10/10	RandomExcursions
4	2	0	0	0	2	0	0	0	2	0.035174	10/10	RandomExcursions
1	1	1	0	1	0	2	1	1	2	0.911413	10/10	RandomExcursions
1	2	2	0	1	1	1	1	0	1	0.911413	10/10	RandomExcursions
1	1	1	0	3	0	1	3	0	0	0.213309	10/10	RandomExcursions
0	3	1	3	0	1	1	1	0	0	0.213309	10/10	RandomExcursions
1	0	1	1	1	2	2	2	0	0	0.739918	10/10	RandomExcursionsVariant
0	2	0	2	1	2	2	0	1	0	0.534146	10/10	RandomExcursionsVariant
1	0	1	0	1	1	3	1	1	1	0.739918	10/10	RandomExcursionsVariant
1	0	1	2	0	1	3	0	1	1	0.534146	10/10	RandomExcursionsVariant
1	1	0	1	1	3	1	0	2	0	0.534146	10/10	RandomExcursionsVariant
1	0	1	2	1	1	0	1	1	2	0.911413	10/10	RandomExcursionsVariant
2	1	1	0	2	1	0	1	1	1	0.911413	10/10	RandomExcursionsVariant
3	0	1	0	1	1	0	0	1	3	0.213309	10/10	RandomExcursionsVariant
3	2	0	0	0	1	2	0	1	1	0.350485	9/10	RandomExcursionsVariant
2	0	1	2	1	0	2	0	1	1	0.739918	10/10	RandomExcursionsVariant
1	1	1	1	4	1	1	0	0	0	0.213309	10/10	RandomExcursionsVariant
2	0	2	0	4	0	0	1	1	0	0.066882	10/10	RandomExcursionsVariant
2	0	1	0	1	2	2	0	1	1	0.739918	10/10	RandomExcursionsVariant
1	2	0	1	2	1	0	3	0	0	0.350485	10/10	RandomExcursionsVariant

1	2	1	3	1	0	1	0	1	0	0.534146	10/10	RandomExcursionsVariant
0	2	2	2	1	1	2	0	0	0	0.534146	10/10	RandomExcursionsVariant
0	0	1	5	0	1	0	3	0	0	0.002043	10/10	RandomExcursionsVariant
0	1	1	2	0	2	1	0	0	3	0.350485	10/10	RandomExcursionsVariant
2	0	1	2	0	1	1	1	1	1	0.911413	10/10	Serial
1	1	2	2	0	0	2	0	0	2	0.534146	10/10	Serial
3	1	0	1	0	1	0	2	0	2	0.350485	10/10	LinearComplexity

- -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 8 for a
sample size = 10 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 8 for a sample size = 10 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- -

A14
Source: ESP32
Suite : NIST
Amount of data: 120000000 octets

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

---

generator is </home/peter/projects/data/esp32.dat>

C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 P-VALUE PROPORTION STATISTICAL TEST

1	1	0	2	1	0	1	0	1	3	0.534146	10/10	Frequency
3	0	1	2	0	0	2	0	0	2	0.213309	10/10	BlockFrequency
1	1	1	0	0	1	1	1	3	1	0.739918	10/10	CumulativeSums
1	0	1	1	0	1	2	1	2	1	0.911413	10/10	CumulativeSums
2	0	1	0	1	0	1	1	3	1	0.534146	9/10	Runs
0	1	0	1	0	2	1	4	1	0	0.122325	10/10	LongestRun
1	1	1	1	1	1	1	3	0	0	0.739918	10/10	Rank
2	0	1	1	2	1	0	1	2	0	0.739918	10/10	FFT
1	1	1	1	1	1	0	1	2	1	0.991468	10/10	NonOverlappingTemplate
1	0	1	2	1	0	1	1	0	3	0.534146	10/10	NonOverlappingTemplate
0	2	1	1	0	2	0	0	2	2	0.534146	10/10	NonOverlappingTemplate

1	0	2	2	1	0	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	4	2	1	1	1	0	0	1	0	0.122325	10/10	NonOverlappingTemplate
2	1	1	0	0	2	3	0	1	0	0.350485	10/10	NonOverlappingTemplate
4	0	1	2	0	1	1	1	0	0	0.122325	10/10	NonOverlappingTemplate
0	2	1	1	0	1	1	1	3	0	0.534146	10/10	NonOverlappingTemplate
2	0	1	1	1	1	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
1	0	3	0	1	2	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
1	1	1	0	2	3	1	0	0	1	0.534146	10/10	NonOverlappingTemplate
2	1	0	0	2	0	2	0	2	1	0.534146	10/10	NonOverlappingTemplate
0	1	1	1	2	2	2	1	0	0	0.739918	10/10	NonOverlappingTemplate
3	1	0	1	0	0	1	0	2	2	0.350485	10/10	NonOverlappingTemplate
0	1	1	1	2	2	1	1	0	1	0.911413	10/10	NonOverlappingTemplate
0	0	1	0	2	1	2	3	1	0	0.350485	10/10	NonOverlappingTemplate
1	0	2	0	2	1	2	1	0	1	0.739918	10/10	NonOverlappingTemplate
0	1	0	0	0	1	2	3	2	1	0.350485	10/10	NonOverlappingTemplate
1	0	0	1	0	0	1	3	3	1	0.213309	10/10	NonOverlappingTemplate
1	2	1	0	0	2	1	0	1	2	0.739918	10/10	NonOverlappingTemplate
3	0	2	0	1	1	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
1	1	2	0	2	0	1	0	0	3	0.350485	10/10	NonOverlappingTemplate
4	1	1	1	0	0	1	1	1	0	0.213309	10/10	NonOverlappingTemplate
2	2	1	0	0	2	0	1	2	0	0.534146	8/10	NonOverlappingTemplate
1	1	0	1	1	1	1	1	0	3	0.739918	10/10	NonOverlappingTemplate
0	0	2	3	2	0	0	0	2	1	0.213309	10/10	NonOverlappingTemplate
0	1	0	3	0	0	2	1	2	1	0.350485	10/10	NonOverlappingTemplate
0	0	2	1	0	3	3	1	0	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	4	2	0	0	0	2	1	0.066882	10/10	NonOverlappingTemplate
0	0	2	2	1	1	3	0	0	1	0.350485	10/10	NonOverlappingTemplate
0	1	0	2	0	1	3	1	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	3	2	0	3	1	0	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	2	1	0	4	0	1	1	0.122325	10/10	NonOverlappingTemplate
0	3	0	0	2	1	1	1	0	2	0.350485	10/10	NonOverlappingTemplate
0	2	1	0	1	3	1	1	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	1	0	1	0	1	2	0	3	0.534146	9/10	NonOverlappingTemplate
2	0	0	0	0	2	1	1	1	3	0.350485	9/10	NonOverlappingTemplate
1	1	1	0	1	0	2	3	0	1	0.534146	9/10	NonOverlappingTemplate
1	1	1	1	3	1	1	1	0	0	0.739918	10/10	NonOverlappingTemplate
2	1	1	0	1	1	1	2	0	1	0.911413	10/10	NonOverlappingTemplate
2	0	1	2	0	0	1	1	1	2	0.739918	10/10	NonOverlappingTemplate
2	4	0	1	0	0	1	2	0	0	0.066882	8/10	NonOverlappingTemplate
0	1	1	1	1	2	0	1	2	1	0.911413	10/10	NonOverlappingTemplate
2	0	0	1	0	1	0	3	1	2	0.350485	9/10	NonOverlappingTemplate
1	1	4	1	2	0	0	0	1	0	0.122325	10/10	NonOverlappingTemplate
1	2	0	1	2	2	0	1	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	1	3	0	0	2	2	1	1	0.350485	10/10	NonOverlappingTemplate
1	1	0	3	0	0	0	2	1	2	0.350485	10/10	NonOverlappingTemplate
3	1	1	1	0	1	2	1	0	0	0.534146	10/10	NonOverlappingTemplate
1	2	1	2	0	2	0	1	1	0	0.739918	10/10	NonOverlappingTemplate
2	2	0	0	1	0	1	3	0	1	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	0	4	1	0	1	1	0.213309	10/10	NonOverlappingTemplate
3	1	2	1	1	1	0	0	0	1	0.534146	9/10	NonOverlappingTemplate
0	2	1	1	1	1	3	0	0	1	0.534146	10/10	NonOverlappingTemplate

1	0	0	1	2	0	2	2	2	0	0.534146	10/10	NonOverlappingTemplate
1	0	1	1	2	1	1	2	1	0	0.911413	10/10	NonOverlappingTemplate
1	1	0	1	1	1	0	0	3	2	0.534146	10/10	NonOverlappingTemplate
1	2	1	1	0	2	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
1	1	1	0	2	1	1	2	1	0	0.911413	10/10	NonOverlappingTemplate
2	0	1	0	1	1	0	1	1	3	0.534146	10/10	NonOverlappingTemplate
0	0	1	1	1	1	2	3	0	1	0.534146	10/10	NonOverlappingTemplate
1	0	1	2	2	0	1	1	0	2	0.739918	9/10	NonOverlappingTemplate
0	0	0	0	2	2	3	1	2	0	0.213309	10/10	NonOverlappingTemplate
1	1	0	3	0	1	1	1	2	0	0.534146	10/10	NonOverlappingTemplate
1	1	1	2	2	0	1	0	0	2	0.739918	10/10	NonOverlappingTemplate
2	3	0	2	0	1	1	1	0	0	0.350485	9/10	NonOverlappingTemplate
1	2	2	0	2	0	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
1	1	0	1	1	2	1	0	1	2	0.911413	9/10	NonOverlappingTemplate
0	1	1	2	1	0	2	2	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	3	0	0	2	1	0	2	0	0.350485	9/10	NonOverlappingTemplate
1	1	1	2	1	0	1	0	2	1	0.911413	10/10	NonOverlappingTemplate
1	2	1	2	0	0	1	0	2	1	0.739918	10/10	NonOverlappingTemplate
2	0	1	1	2	1	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
2	1	2	1	0	0	2	0	2	0	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	1	1	0	1	2	1	0.991468	10/10	NonOverlappingTemplate
0	1	1	1	2	2	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
0	2	0	0	2	1	1	0	1	3	0.350485	10/10	NonOverlappingTemplate
0	2	2	1	2	0	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	0	0	1	1	3	1	0	2	2	0.350485	10/10	NonOverlappingTemplate
0	0	1	2	0	2	2	0	0	3	0.213309	10/10	NonOverlappingTemplate
2	1	1	0	1	0	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	3	2	2	2	0	0	0	0	0	0.213309	10/10	NonOverlappingTemplate
2	0	0	0	1	1	0	2	2	2	0.534146	9/10	NonOverlappingTemplate
0	2	3	0	0	1	1	0	3	0	0.122325	10/10	NonOverlappingTemplate
1	0	0	0	3	1	2	2	0	1	0.350485	10/10	NonOverlappingTemplate
0	2	1	1	1	2	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
2	1	0	1	0	2	0	0	2	2	0.534146	10/10	NonOverlappingTemplate
0	0	0	1	3	1	1	1	3	0	0.213309	10/10	NonOverlappingTemplate
4	0	1	0	0	0	2	1	0	2	0.066882	10/10	NonOverlappingTemplate
1	2	0	0	0	0	1	3	3	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	1	3	0	0	2	0	3	0.122325	10/10	NonOverlappingTemplate
0	1	1	0	2	0	0	2	2	2	0.534146	10/10	NonOverlappingTemplate
1	2	0	0	0	3	0	0	0	4	0.017912	10/10	NonOverlappingTemplate
0	1	1	0	2	1	0	1	1	3	0.534146	10/10	NonOverlappingTemplate
1	0	0	2	1	2	1	1	2	0	0.739918	10/10	NonOverlappingTemplate
0	1	0	1	2	0	1	3	2	0	0.350485	10/10	NonOverlappingTemplate
1	1	0	1	0	1	4	1	1	0	0.213309	9/10	NonOverlappingTemplate
2	2	1	0	1	1	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	1	1	0	1	0	3	2	1	0	0.534146	10/10	NonOverlappingTemplate
0	0	2	1	1	1	0	3	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	3	1	1	0	1	2	0	1	0.534146	10/10	NonOverlappingTemplate
0	1	2	1	2	0	0	0	2	2	0.534146	10/10	NonOverlappingTemplate
1	1	2	1	2	1	0	1	1	0	0.911413	10/10	NonOverlappingTemplate
1	2	0	0	1	2	2	1	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	0	0	1	1	1	2	3	2	0.350485	10/10	NonOverlappingTemplate

1	4	0	1	0	0	0	2	1	1	0.122325	10/10	NonOverlappingTemplate
1	0	1	4	0	1	0	0	1	2	0.122325	10/10	NonOverlappingTemplate
2	0	1	1	1	1	1	2	0	1	0.911413	10/10	NonOverlappingTemplate
1	2	0	1	1	2	1	0	1	1	0.911413	10/10	NonOverlappingTemplate
0	2	1	1	2	0	0	2	1	1	0.739918	10/10	NonOverlappingTemplate
2	0	2	0	0	1	0	0	1	4	0.066882	10/10	NonOverlappingTemplate
1	2	0	2	1	0	2	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	0	1	3	1	1	1	0	2	1	0.534146	10/10	NonOverlappingTemplate
1	1	0	0	0	3	0	1	2	2	0.350485	10/10	NonOverlappingTemplate
1	1	0	1	3	0	0	1	2	1	0.534146	10/10	NonOverlappingTemplate
0	2	1	1	1	2	2	0	1	0	0.739918	10/10	NonOverlappingTemplate
0	1	1	0	1	2	2	0	0	3	0.350485	10/10	NonOverlappingTemplate
0	1	3	1	2	0	1	1	1	0	0.534146	10/10	NonOverlappingTemplate
1	3	2	0	0	1	2	0	0	1	0.350485	10/10	NonOverlappingTemplate
1	2	2	0	1	0	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	0	1	1	1	1	1	2	0	2	0.911413	10/10	NonOverlappingTemplate
0	1	1	0	0	2	2	3	1	0	0.350485	10/10	NonOverlappingTemplate
1	2	0	2	2	0	0	2	0	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	2	1	2	1	3	0	0	0.350485	10/10	NonOverlappingTemplate
0	0	3	0	1	1	0	2	2	1	0.350485	10/10	NonOverlappingTemplate
2	1	3	0	0	0	3	0	0	1	0.122325	10/10	NonOverlappingTemplate
1	0	1	1	2	1	1	1	2	0	0.911413	10/10	NonOverlappingTemplate
2	3	1	0	0	0	0	2	0	2	0.213309	8/10	NonOverlappingTemplate
1	1	1	1	1	0	1	0	2	2	0.911413	10/10	NonOverlappingTemplate
4	1	0	1	0	2	0	0	1	1	0.122325	10/10	NonOverlappingTemplate
0	1	0	0	2	2	0	3	2	0	0.213309	10/10	NonOverlappingTemplate
0	0	2	0	3	3	0	0	0	2	0.066882	10/10	NonOverlappingTemplate
2	0	0	2	1	0	1	0	1	3	0.350485	10/10	NonOverlappingTemplate
1	0	2	2	0	1	0	0	3	1	0.350485	10/10	NonOverlappingTemplate
1	0	0	3	3	1	0	0	2	0	0.122325	10/10	NonOverlappingTemplate
2	1	1	0	0	3	2	0	1	0	0.350485	10/10	NonOverlappingTemplate
1	0	0	1	0	1	4	2	1	0	0.122325	10/10	NonOverlappingTemplate
3	0	0	1	3	0	0	1	1	1	0.213309	10/10	NonOverlappingTemplate
3	1	1	1	0	1	2	0	0	1	0.534146	10/10	NonOverlappingTemplate
0	0	1	1	3	0	1	2	0	2	0.350485	10/10	NonOverlappingTemplate
2	0	0	1	0	0	1	3	2	1	0.350485	10/10	NonOverlappingTemplate
1	1	1	1	1	0	0	2	3	0	0.534146	10/10	NonOverlappingTemplate
1	0	1	4	0	2	0	0	2	0	0.066882	10/10	NonOverlappingTemplate
4	1	1	0	0	0	0	3	1	0	0.035174	10/10	NonOverlappingTemplate
2	1	0	1	1	1	1	2	1	0	0.911413	10/10	NonOverlappingTemplate
0	0	1	1	2	3	0	1	2	0	0.350485	10/10	NonOverlappingTemplate
3	0	1	3	0	0	0	1	0	2	0.122325	10/10	NonOverlappingTemplate
2	1	2	1	0	0	2	0	2	0	0.534146	10/10	NonOverlappingTemplate
8	0	1	0	1	0	0	0	0	0	0.000000 *	7/10	* OverlappingTemplate
2	0	0	2	0	2	0	1	3	0	0.213309	10/10	Universal
1	2	0	1	2	1	1	0	1	1	0.911413	10/10	ApproximateEntropy
0	2	2	0	0	1	2	0	2	0	—-	9/9	RandomExcursions
4	0	2	0	0	0	0	0	0	3	—-	8/9	RandomExcursions
4	0	0	1	0	0	0	1	1	2	—-	9/9	RandomExcursions
1	0	0	2	1	1	0	2	1	1	—-	9/9	RandomExcursions
1	2	1	0	1	1	2	1	0	0	—-	9/9	RandomExcursions

1	1	1	2	1	1	0	1	0	1	—-	9/9	RandomExcursions
1	1	1	0	3	0	0	1	0	2	—-	9/9	RandomExcursions
0	1	1	0	1	0	2	2	2	0	—-	9/9	RandomExcursions
1	0	1	1	0	0	2	1	3	0	—-	9/9	RandomExcursionsVariant
0	1	0	1	1	1	2	1	1	1	—-	9/9	RandomExcursionsVariant
0	1	0	0	1	3	0	2	0	2	—-	9/9	RandomExcursionsVariant
1	0	0	2	0	1	1	1	2	1	—-	9/9	RandomExcursionsVariant
0	1	0	3	1	1	0	0	3	0	—-	9/9	RandomExcursionsVariant
0	1	3	0	1	0	1	0	2	1	—-	9/9	RandomExcursionsVariant
1	1	1	1	0	0	1	1	2	1	—-	9/9	RandomExcursionsVariant
0	1	1	2	0	1	0	0	2	2	—-	9/9	RandomExcursionsVariant
0	0	1	1	2	0	0	3	0	2	—-	9/9	RandomExcursionsVariant
0	1	1	1	1	0	4	0	1	0	—-	9/9	RandomExcursionsVariant
0	1	0	1	0	3	0	3	0	1	—-	9/9	RandomExcursionsVariant
0	1	0	1	0	1	1	2	0	3	—-	9/9	RandomExcursionsVariant
0	1	1	0	1	0	1	3	0	2	—-	9/9	RandomExcursionsVariant
0	0	1	2	1	1	0	2	0	2	—-	9/9	RandomExcursionsVariant
0	0	1	1	1	1	1	1	1	2	—-	9/9	RandomExcursionsVariant
0	0	1	0	2	2	0	2	2	0	—-	9/9	RandomExcursionsVariant
0	0	0	2	2	1	2	1	1	0	—-	9/9	RandomExcursionsVariant
0	0	0	1	2	0	3	1	1	1	—-	9/9	RandomExcursionsVariant
0	0	4	0	0	2	1	2	0	1	0.066882	10/10	Serial
1	1	0	1	1	0	1	1	3	1	0.739918	10/10	Serial
0	2	1	0	3	0	2	0	1	1	0.350485	10/10	LinearComplexity

- -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 8 for a
sample size = 10 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 8 for a sample size = 9 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- -