Blum Blum Shub (B.B.S.) is a pseudorandom number generator proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub.

Blum Blum Shub takes the form:

\[ x_{n+1} = x_n^2 \mod M \]

where \( M = pq \) is the product of two large prime number \( p \) and \( q \). At each step of the algorithm, some output is derived from \( x_{n+1} \); the output is commonly either the parity bit of \( x_{n+1} \) or one or more of the least significant bits of \( x_{n+1} \).

The seed \( x_0 \) should be an integer that’s not 1 or co-prime to \( M \) (i.e. \( p \) and \( q \) are not factors of \( x_0 \)).

The two primes, \( p \) and \( q \), should both be congruent to 3 (mod 4) (this guarantees that each quadratic residue has one square root which is also a quadratic residue) and \( \gcd(\text{Euler's totient function} \mid (p-1), (q-1)) \) should be small (this makes the cycle length large).

An interesting characteristic of the Blum Blum Shub generator is the possibility to calculate any \( x_i \) value directly (via Euler's Theorem):

\[ x_i = (x_0^{2^i \mod \varphi(M)}) \mod M \]

where \( \varphi \) is the Carmichael function. (Here we have \( \varphi = \varphi(pq) = \text{lcm}(p-1, q-1) \)).

## Code

Here we have a simple implementation of BBS algorithm:

**Input parameters:**

\( p, q \) & seed. They need to be assigned on the way told above to initialize the bbs class before start asking for random numbers.

**Functions:**

getrandom(), which returns the \( x_{n+1} \) random number of the bbs sequence and updates the index.

getirandom(int i), which returns the \( x_i \) random number of the bbs sequence specified by the input parameter \( i \)(second algorithm). Both functions return a floating point number in the range \([0,1)\).

**Main:**
Inside the main class we create an instance of a bbs class with the parameters specified above. Then we print the first n numbers of this bbs sequence.

**java code**

```java
class bbs {
    private double p, q, M, seed, actual;
    public bbs(double p, double q, double seed) {
        this.p = p;
        this.q = q;
        this.M = p*q;
        this.seed = seed;
        this.actual = seed;
    }

    public double getrandom() {
        double r = actual*actual%M;
        actual = r;
        return r/M;
    }

    private double gcd(double a, double b) {
        if(b == 0) return a;
        return gcd(b, a%b);
    }

    public double getirandom(int i) {
        double g = gcd(p, q);
        double l = (p-1)*(q-1)/g;
        double exp = 1;
        for(int j = 1; j <= i; ++j) exp = exp*2%l;
        double x0 = seed*seed;
        double r = x0;
        for(double j = 2; j<=exp; ++j) {
            r = r*x0%M;
        }
        return r/M;
    }
}

class Main {
    public static void main(String[] args) {
        double p = 11;
        double q = 19;
        double seed = 3;
        bbs b = new bbs(p,q,seed);
        System.out.println(1000);
        for(int i = 0; i <= 1000; ++i) {
            System.out.println(b.getrandom());
        }
    }
}
```
c++ code

```cpp
#include <iostream>
#include <cmath>
using namespace std;
typedef long double ll;

class bbs{
    ll p, q, M, seed, actual;

    ll gcd(ll a, ll b){
        if(b == 0) return a;
        return gcd(b, fmod(a,b));
    }

public:
    bbs(ll p, ll q, ll s){
        this->p = p;
        this->q = q;
        this->seed = s;
        M = p*q;
        actual = s;
    }

    ll getrandom(){
        ll r = fmod(actual*actual,M);
        actual = r;
        return r;
    }

    ll getirandom(int i){
        ll g = gcd(p, q);
        ll lcm = p*q/g;
        ll exp = 1;
        for(int j = 1; j <= i; ++j) exp = fmod((exp+exp),lcm);
        ll x0 = seed*seed;
        ll r = x0;
        for(int j = 2; j <= exp; ++j) r = fmod((r*x0),M);
        return r;
    }
};

int main(){
    ll p = 11;
    ll q = 19;
    ll s = 3;
}
```

java code

```
public class Blum_Blum_Shub{

    public static void main(String[] args){
        long p = 11;
        long q = 19;
        long s = 3;
        // Code here...
    }
}
```
bbs b(p, q, s);
int n = 1000;
cout << n << endl;
for(int i = 0; i < n; ++i) {
    cout << b.getrandom()/M << endl;
}

Parameterizations

Parameterization 1:

p = 15485867
q = 23878409
seed = 8353

Parameterization 2:

p = 31415821
q = 772531
seed = 1878892440

Parameterization 3:

p = 3264011
q = 472990951
seed = 131893

Parameterization 4:

p = 3263849
q = 1302498943
seed = 6367859

Parameterization 5:

p = 3264011
q = 472990951
seed = 9883

Parameterization 6:

p = 4990364893
q = 58105693
seed = 6681193

Parameterization 7:

p = 20359
q = 43063
seed = 1993

Parameterization 8:

p = 87566873
q = 15485143
seed = 193945
Parameterization 9:

\[ p = 87566873 \]
\[ q = 15485143 \]
\[ \text{seed} = 740191 \]

Parameterization 10:

\[ p = 87566873 \]
\[ q = 5631179 \]
\[ \text{seed} = 191 \]

Tests

<table>
<thead>
<tr>
<th>Parameterization</th>
<th>FREQUENCIES</th>
<th>SERIE</th>
<th>GAP</th>
<th>YULE-WALKER</th>
<th>DISTANCES</th>
<th>AUTOCORRELACION</th>
<th>KOLMOGOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(47) no(3)</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(47) no(3)</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(47) no(3)</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes(48) no(2)</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(44) no(6)</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(46) no(4)</td>
<td>yes</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(43) no(7)</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(48) no(2)</td>
<td>yes</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(45) no(5)</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes(46) no(4)</td>
<td>yes</td>
</tr>
</tbody>
</table>