Assignment-1

Abheek Ghosh

140123047

January 22, 2016

A custom header file with important functions. Code for C++

```
#include<iostream>
#include<iomanip>

using namespace std;

//Our main uniform random values genrator class
class UniformGenerator {
  public:
    UniformGenerator() {
      a=1,b=0,m=1;   //Just some initializing values.
    }
    unsigned long long int a,b,m;

    //The first value is the value of x0 and second no of terms in sequence.
    int generate(unsigned long int x, unsigned long int n) {
      long double u;
      long rep=x,k;
      cout<<"Initial value, $x_0$ = "<<x<<"\n\\begin{center} \\begin{tabular}{||c | c | c||}
          \\hline\n";
      cout<<"$i$ index & $x$ & $u$ \\\\ [0.5ex] \\hline \\hline";
      for(int j=0;j<n;++j) {
        u = (long double)x/(long double)m;
        cout<<j<<" & "<<x<<" & "<<setprecision(10)<<u<<"\\\\\n \\hline \n";
        x = (a*x + b) % m;
        if(rep == x) {
          k=j+1;
          rep = -1;
        }
      }
      cout<<"\\end{tabular} \n \\end{center}";
      cout<<" Repetition after "<<k<<" terms.\n\\\\ \\noindent\\rule[0.5ex]{\\linewidth}{1pt}\
          n";
      return 0;
```

```
32      }
33       int covariance(unsigned long int x, unsigned long int n) {
34        long double u,v;
35        cout<<"Initial value, $x_0$ = "<<x<<"\n";
36          u = (long double)x/(long double)m;
37        for(int j=0;j<n;++j) {
38          x = (a*x + b) % m;
39          v = u;
40          u = (long double)x/(long double)m;
41          cout<<"u"<<j<<"= "<<setprecision(10)<<v<<"\tu"<<j+1<<"= "<<u<<"\n";
42        }
43        cout<<endl;
44        return 0;
45      }
46      //The first value is the value of x0, second no of terms in sequence
47      //and third is the no. of intervals.
48      int goodness(unsigned long long int x, unsigned long long int n, int k) {
49        long double u;
50        // Initializing the density array
51        unsigned long long Density[k];
52        for(int i=0;i<k;++i) {
53          Density[i] = 0;
54        }
55        //Generating terms
56        for(int j=0;j<n;++j) {
57          u = (long double)x/(long double)m;
58          Density[(int)(u*k)]++ ;
59          x = (a*x + b) % m;
60        }
61        //Printing frequency output
62        u = (long double)1/(long double)k;
63        cout<<"\n\\begin{center} \\begin{tabular}{||c | c||}  \\hline\n";
64        cout<<"Range & Frequency \\\\ [0.5ex] \\hline \\hline";
65        for(int i=0;i<k;++i) {
66          cout<<setprecision(2)<<(u*i)<<"-"<<(u*(i+1))<<" & "<<Density[i]<<"\\\\\n \\hline \n";
67         // cout<<"\t"<<setprecision(20)<<((long double)Density[i]/n)<<endl;
68        }
69        cout<<"\\end{tabular} \n \\end{center}\n";
70        return 0;
71      }
72    };
```

1 Question 1

For $x = 0 \ mod \ m$ we get only 0 for x. But for $x \neq 0 \ mod \ m$ we get 10 distinct values for a=6 and 5 distinct values for a=3. So a=6 is better selection. The result has been tabulated below.

Code for C++

```
#include <iostream>
#include <iomanip>
#include "uniform.h" //Main file with functions

using namespace std;

int main() {
  UniformGenerator uni;
  //The values of variables
  cout << "Enter a, b and m respectively.\n";
  cin>> uni.a >> uni.b >> uni.m;
  for(int i=0;i<11;++i)
    uni.generate(i,12);

  return 0;
}
```

$$a = 6, b = 0, m = 11$$

Initial value, $x_0 = 0$

i index	x	u
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

Repetition after 1 terms.

Initial value, $x_0 = 1$

i index	x	u
0	1	0.09090909091
1	6	0.5454545455
2	3	0.2727272727
3	7	0.6363636364
4	9	0.8181818182
5	10	0.9090909091
6	5	0.4545454545
7	8	0.7272727273
8	4	0.3636363636
9	2	0.1818181818
10	1	0.09090909091
11	6	0.5454545455

Repetition after 10 terms.

Initial value, $x_0 = 2$

i index	x	u
0	2	0.1818181818
1	1	0.09090909091
2	6	0.5454545455
3	3	0.2727272727
4	7	0.6363636364
5	9	0.8181818182
6	10	0.9090909091
7	5	0.4545454545
8	8	0.7272727273
9	4	0.3636363636
10	2	0.1818181818
11	1	0.09090909091

Repetition after 10 terms.

Initial value, $x_0 = 3$

i index	x	u
0	3	0.2727272727
1	7	0.6363636364
2	9	0.8181818182
3	10	0.9090909091
4	5	0.4545454545
5	8	0.7272727273
6	4	0.3636363636
7	2	0.1818181818
8	1	0.09090909091
9	6	0.5454545455
10	3	0.2727272727
11	7	0.6363636364

Repetition after 10 terms.

Initial value, $x_0 = 4$

i index	x	u
0	4	0.3636363636
1	2	0.1818181818
2	1	0.09090909091
3	6	0.5454545455
4	3	0.2727272727
5	7	0.6363636364
6	9	0.8181818182
7	10	0.9090909091
8	5	0.4545454545
9	8	0.7272727273
10	4	0.3636363636
11	2	0.1818181818

Repetition after 10 terms.

Initial value, $x_0 = 5$

i index	x	u
0	5	0.4545454545
1	8	0.7272727273
2	4	0.3636363636
3	2	0.1818181818
4	1	0.09090909091
5	6	0.5454545455
6	3	0.2727272727
7	7	0.6363636364
8	9	0.8181818182
9	10	0.9090909091
10	5	0.4545454545
11	8	0.7272727273

Repetition after 10 terms.

Initial value, $x_0 = 6$

i index	x	u
0	6	0.5454545455
1	3	0.2727272727
2	7	0.6363636364
3	9	0.8181818182
4	10	0.9090909091
5	5	0.4545454545
6	8	0.7272727273
7	4	0.3636363636
8	2	0.1818181818
9	1	0.09090909091
10	6	0.5454545455
11	3	0.2727272727

Repetition after 10 terms.

Initial value, $x_0 = 7$

i index	x	u
0	7	0.6363636364
1	9	0.8181818182
2	10	0.9090909091
3	5	0.4545454545
4	8	0.7272727273
5	4	0.3636363636
6	2	0.1818181818
7	1	0.09090909091
8	6	0.5454545455
9	3	0.2727272727
10	7	0.6363636364
11	9	0.8181818182

Repetition after 10 terms.

Initial value, $x_0 = 8$

i index	x	u
0	8	0.7272727273
1	4	0.3636363636
2	2	0.1818181818
3	1	0.09090909091
4	6	0.5454545455
5	3	0.2727272727
6	7	0.6363636364
7	9	0.8181818182
8	10	0.9090909091
9	5	0.4545454545
10	8	0.7272727273
11	4	0.3636363636

Repetition after 10 terms.

Initial value, $x_0 = 9$

i index	x	u
0	9	0.8181818182
1	10	0.9090909091
2	5	0.4545454545
3	8	0.7272727273
4	4	0.3636363636
5	2	0.1818181818
6	1	0.09090909091
7	6	0.5454545455
8	3	0.2727272727
9	7	0.6363636364
10	9	0.8181818182
11	10	0.9090909091

Repetition after 10 terms.

Initial value, $x_0 = 10$

i index	x	u
0	10	0.9090909091
1	5	0.4545454545
2	8	0.7272727273
3	4	0.3636363636
4	2	0.1818181818
5	1	0.09090909091
6	6	0.5454545455
7	3	0.2727272727
8	7	0.6363636364
9	9	0.8181818182
10	10	0.9090909091
11	5	0.4545454545

Repetition after 10 terms.

$$a = 3, b = 0, m = 11$$

Initial value, $x_0 = 0$

i index	x	u
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

Repetition after 1 terms.

Initial value, $x_0 = 1$

i index	x	u
0	1	0.09090909091
1	3	0.2727272727
2	9	0.8181818182
3	5	0.4545454545
4	4	0.3636363636
5	1	0.09090909091
6	3	0.2727272727
7	9	0.8181818182
8	5	0.4545454545
9	4	0.3636363636
10	1	0.09090909091
11	3	0.2727272727

Repetition after 5 terms.

Initial value, $x_0 = 2$

i index	x	u
0	2	0.1818181818
1	6	0.5454545455
2	7	0.6363636364
3	10	0.9090909091
4	8	0.7272727273
5	2	0.1818181818
6	6	0.5454545455
7	7	0.6363636364
8	10	0.9090909091
9	8	0.7272727273
10	2	0.1818181818
11	6	0.5454545455

Repetition after 5 terms.

Initial value, $x_0 = 3$

i index	x	u
0	3	0.2727272727
1	9	0.8181818182
2	5	0.4545454545
3	4	0.3636363636
4	1	0.09090909091
5	3	0.2727272727
6	9	0.8181818182
7	5	0.4545454545
8	4	0.3636363636
9	1	0.09090909091
10	3	0.2727272727
11	9	0.8181818182

Repetition after 5 terms.

Initial value, $x_0 = 4$

i index	x	u
0	4	0.3636363636
1	1	0.09090909091
2	3	0.2727272727
3	9	0.8181818182
4	5	0.4545454545
5	4	0.3636363636
6	1	0.09090909091
7	3	0.2727272727
8	9	0.8181818182
9	5	0.4545454545
10	4	0.3636363636
11	1	0.09090909091

Repetition after 5 terms.

Initial value, $x_0 = 5$

i index	x	u
0	5	0.4545454545
1	4	0.3636363636
2	1	0.09090909091
3	3	0.2727272727
4	9	0.8181818182
5	5	0.4545454545
6	4	0.3636363636
7	1	0.09090909091
8	3	0.2727272727
9	9	0.8181818182
10	5	0.4545454545
11	4	0.3636363636

Repetition after 5 terms.

Initial value, $x_0 = 6$

i index	x	u
0	6	0.5454545455
1	7	0.6363636364
2	10	0.9090909091
3	8	0.7272727273
4	2	0.1818181818
5	6	0.5454545455
6	7	0.6363636364
7	10	0.9090909091
8	8	0.7272727273
9	2	0.1818181818
10	6	0.5454545455
11	7	0.6363636364

Repetition after 5 terms.

Initial value, $x_0 = 7$

i index	x	u
0	7	0.6363636364
1	10	0.9090909091
2	8	0.7272727273
3	2	0.1818181818
4	6	0.5454545455
5	7	0.6363636364
6	10	0.9090909091
7	8	0.7272727273
8	2	0.1818181818
9	6	0.5454545455
10	7	0.6363636364
11	10	0.9090909091

Repetition after 5 terms.

Initial value, $x_0 = 8$

i index	x	u
0	8	0.7272727273
1	2	0.1818181818
2	6	0.5454545455
3	7	0.6363636364
4	10	0.9090909091
5	8	0.7272727273
6	2	0.1818181818
7	6	0.5454545455
8	7	0.6363636364
9	10	0.9090909091
10	8	0.7272727273
11	2	0.1818181818

Repetition after 5 terms.

Initial value, $x_0 = 9$

i index	x	u
0	9	0.8181818182
1	5	0.4545454545
2	4	0.3636363636
3	1	0.09090909091
4	3	0.2727272727
5	9	0.8181818182
6	5	0.4545454545
7	4	0.3636363636
8	1	0.09090909091
9	3	0.2727272727
10	9	0.8181818182
11	5	0.4545454545

Repetition after 5 terms.

Initial value, $x_0 = 10$

i index	x	u
0	10	0.9090909091
1	8	0.7272727273
2	2	0.1818181818
3	6	0.5454545455
4	7	0.6363636364
5	10	0.9090909091
6	8	0.7272727273
7	2	0.1818181818
8	6	0.5454545455
9	7	0.6363636364
10	10	0.9090909091
11	8	0.7272727273

Repetition after 5 terms.

2 Question 2

We observe that good random numbers are generated with the given values. The number distribute more uniformly with increase in data, showing probability of a number in each interval is approximately same.

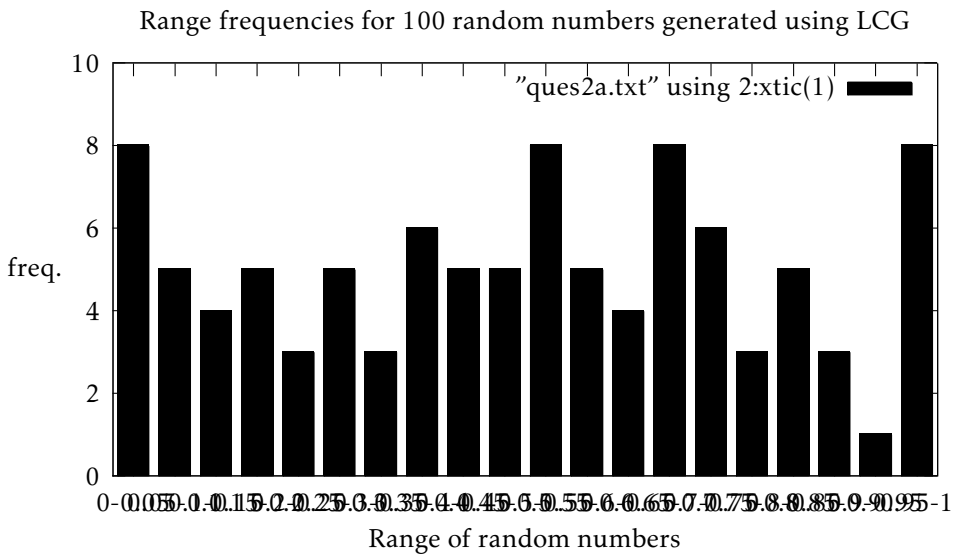
Code for C++

```
#include<iostream>
#include<iomanip>
#include"uniform.h"

using namespace std;

int main() {
  UniformGenerator uni;
  //The values of variables
  uni.m = 244944;
  uni.a = 1597;
  uni.b = 0;
  //Taking x per our choice;
  unsigned long int x = 1;
  cout<<"$$a= "<<uni.a<<",  b= "<<uni.b<<",  m= "<<uni.m<<",  x_0= "<<x<<"$$\n";
  //Number of values to generate.
  unsigned long int n = 100;
  for(int i=0;i<5;++i) {
    cout<<"\n\n$$n= "<<n<<"$$\n";
    uni.goodness(x,n,20);
    n=n*10;
    cout<<"\n\\\\ \\noindent\\rule[0.5ex]{\\linewidth}{1pt}\n\n";
  }
  return 0;
}
```

$$a = 1597, b = 0, m = 244944, x_0 = 1$$

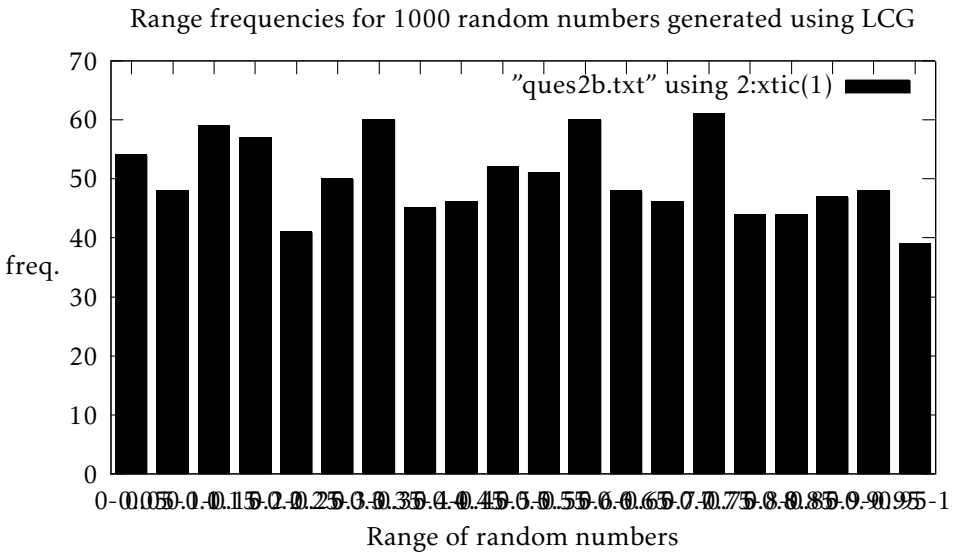
$$n = 100$$

Range	Frequency
0-0.05	8
0.05-0.1	5
0.1-0.15	4
0.15-0.2	5
0.2-0.25	3
0.25-0.3	5
0.3-0.35	3
0.35-0.4	6
0.4-0.45	5
0.45-0.5	5
0.5-0.55	8
0.55-0.6	5
0.6-0.65	4
0.65-0.7	8
0.7-0.75	6
0.75-0.8	3
0.8-0.85	5
0.85-0.9	3
0.9-0.95	1
0.95-1	8


Range frequencies for 100 random numbers generated using LCG
"ques2a.txt" using 2:xtic(1)
freq.
10
8
6
4
2
0
Range of random numbers

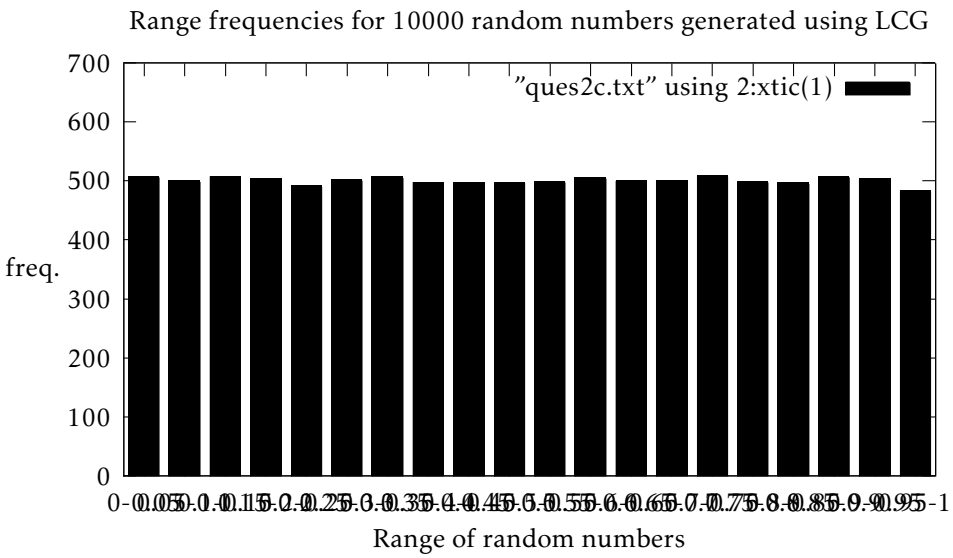
$n = 1000$

Range	Frequency
0-0.05	54
0.05-0.1	48
0.1-0.15	59
0.15-0.2	57
0.2-0.25	41
0.25-0.3	50
0.3-0.35	60
0.35-0.4	45
0.4-0.45	46
0.45-0.5	52
0.5-0.55	51
0.55-0.6	60
0.6-0.65	48
0.65-0.7	46
0.7-0.75	61
0.75-0.8	44
0.8-0.85	44
0.85-0.9	47
0.9-0.95	48
0.95-1	39


Range frequencies for 1000 random numbers generated using LCG
"ques2b.txt" using 2:xtic(1)
freq.
70
60
50
40
30
20
10
0
Range of random numbers

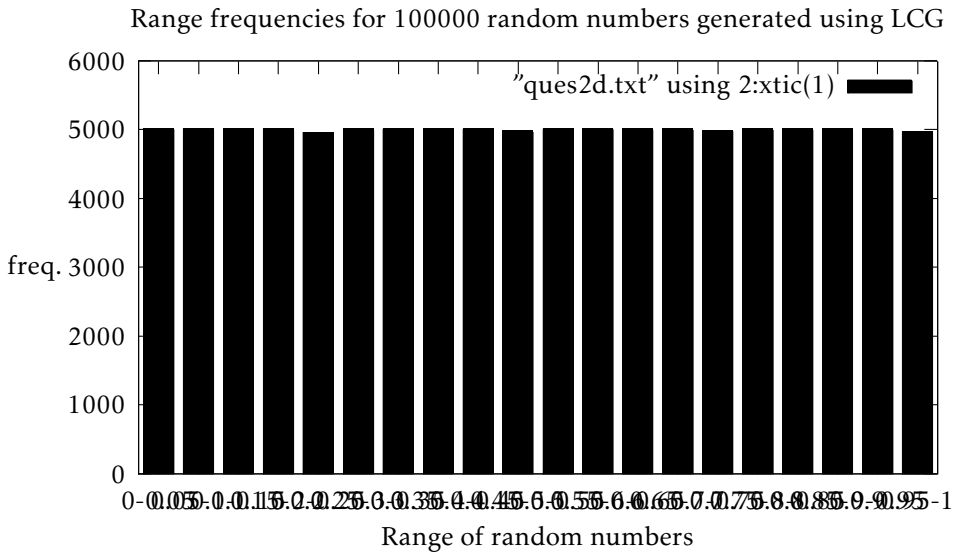
$n = 10000$

Range	Frequency
0-0.05	507
0.05-0.1	499
0.1-0.15	507
0.15-0.2	504
0.2-0.25	490
0.25-0.3	501
0.3-0.35	506
0.35-0.4	497
0.4-0.45	496
0.45-0.5	497
0.5-0.55	498
0.55-0.6	505
0.6-0.65	499
0.65-0.7	500
0.7-0.75	508
0.75-0.8	498
0.8-0.85	496
0.85-0.9	507
0.9-0.95	502
0.95-1	483


Range frequencies for 10000 random numbers generated using LCG
"ques2c.txt" using 2:xtic(1)
700
600
500
400
300
200
100
0
freq.
Range of random numbers

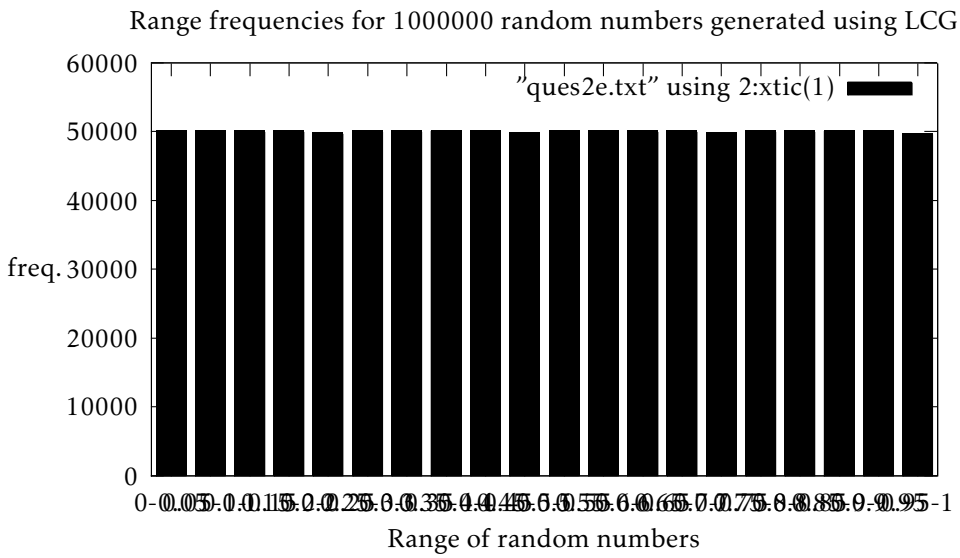
$n = 100000$

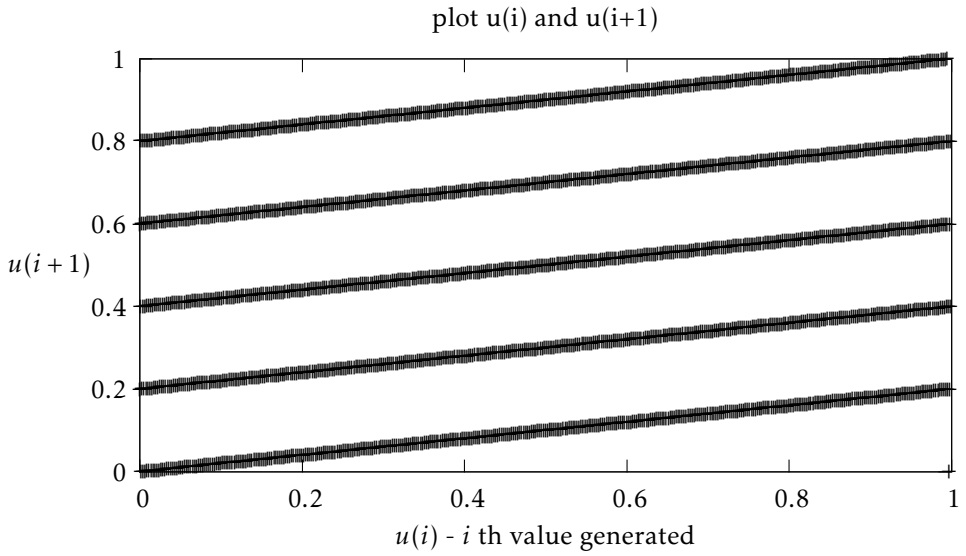
Range	Frequency
0-0.05	5012
0.05-0.1	5006
0.1-0.15	5014
0.15-0.2	5009
0.2-0.25	4962
0.25-0.3	5009
0.3-0.35	5008
0.35-0.4	5003
0.4-0.45	5005
0.45-0.5	4974
0.5-0.55	5007
0.55-0.6	5017
0.6-0.65	5002
0.65-0.7	5005
0.7-0.75	4985
0.75-0.8	5003
0.8-0.85	5000
0.85-0.9	5008
0.9-0.95	5003
0.95-1	4968


Range frequencies for 100000 random numbers generated using LCG
"ques2d.txt" using 2:xtic(1)
6000
5000
4000
freq. 3000
2000
1000
0
Range of random numbers

$n = 1000000$

Range	Frequency
0-0.05	50070
0.05-0.1	50068
0.1-0.15	50070
0.15-0.2	50067
0.2-0.25	49729
0.25-0.3	50067
0.3-0.35	50072
0.35-0.4	50069
0.4-0.45	50068
0.45-0.5	49723
0.5-0.55	50071
0.55-0.6	50071
0.6-0.65	50071
0.65-0.7	50071
0.7-0.75	49726
0.75-0.8	50068
0.8-0.85	50066
0.85-0.9	50073
0.9-0.95	50066
0.95-1	49714


Range frequencies for 1000000 random numbers generated using LCG
"ques2e.txt" using 2:xtic(1)
60000
50000
40000
freq. 30000
20000
10000
0
Range of random numbers



3 Question 3

The values generated are not good. The $i+1$ value depends on i the value. The Autocorrelation is very high, which is not good for randomness. The density of values generated is near 5 lines. To improve this we must select values such that the no. of lines increases and distance between them decreases.

Code for C++

```
#include<iostream>
#include<iomanip>
#include"uniform.h"

using namespace std;

int main() {
  UniformGenerator uni;
  //The values of variables
  uni.m = 2048;
  uni.a = 1229;
  uni.b = 1;
  //Taking x per our choice;
  unsigned long int x = 1;
  //Number of values to generate.
  unsigned long int n = 2047;
  uni.covariance(x,n);
  return 0;
}
```