**SUPROTIK DEY**

**IT Hx-31**

**4TH SEM, 510815050**

1. **Write a program in C/C++/Java to model and simulate the juggling process with n items.**

**SOURCE CODE:**

#include <stdio.h>

#define MAX 100

void printArray(int \*arr, int n)

{

int i;

printf("\nList:\n");

for( i = 0; i < n; i++ )

{

printf("\n%d", arr[i]);

}

}

void juggling(int \*arr, int n)

{

if( n==0 || n==1)

return;

int val, i;

val = arr[n-1];

for( i = n-1; i >= 0; i-- )

{

arr[i+1] = arr[i];

}

arr[0] = val;

}

int main()

{

int n, arr[MAX], i, opt;

printf("\nEnter the number of things you want to juggle:");

scanf("%d", &n);

printf("\nEnter the numbers:\n");

for( i = 0; i < n; i++ )

{

scanf("%d", &arr[i]);

}

while(1)

{

printArray(arr, n);

juggling(arr, n);

printf("\npress any key to continue or x to end...");

if(getchar()=='x')

break;

}

return 0;

}

SAMPLE OUTPUT:

Enter the number of things you want to juggle:5

Enter the numbers:

1

2

3

4

5

List:

1

2

3

4

5

press any key to continue or x to end...

List:

5

1

2

3

4

press any key to continue or x to end...

List:

4

5

1

2

3

press any key to continue or x to end...

x

1. **Write a program in C/C++/Java to model a system that generates a set of n random numbers within the range [a, b], where n, a and b are user inputs. Test the randomness of the generated sequence of numbers.**

**SOURCE CODE: (** Linear congruential generator **)**

#include <stdio.h>

//Linear congruential generator

#define X0 2

#define mul 2

#define c 1

int lcgRand(int a, int b, int \*xn)

{

int m, new;

m = b - a + 1;

if(X0 > m || mul > m || c > m)

{

printf("\nCannot compute!!");

return 0;

}

new = ( mul \* (\*xn) + c) % m;

\*xn = new;

return (new + a);

}

int main()

{

int A, B, N, i;

int Xn = X0;

printf("\nEnter A(the starting value),\nB(the ending value),\nN(the number of values to print):\n");

scanf("%d %d %d", &A, &B, &N);

printf("\nThe random nos.:\n");

for( i = 1; i <= N; i++ )

{

printf("\n%d", lcgRand(A, B, &Xn));

}

return 0;

}

SAMPLE OUTPUTS:

Enter A(the starting value),

B(the ending value),

N(the number of values to print):

3

15

6

The random nos.:

8

14

13

11

7

12

Enter A(the starting value),

B(the ending value),

N(the number of values to print):

2

30

8

The random nos.:

7

13

25

20

10

19

8

15

**SOURCE CODE:** ( Blum Blum Shub generator )

//blum blum shub generator..

#include <stdio.h>

#define P 47

#define Q 67

#define R 207

#define maxlen 100

int gcd(int a, int b)

{

// Everything divides 0

if (a == 0 || b == 0)

return 0;

// base case

if (a == b)

return a;

// a is greater

if (a > b)

return gcd(a-b, b);

return gcd(a, b-a);

}

int main()

{

int N, X, A, B, iter, stream[maxlen], i, M;

N = P\*Q;

if((P%4!=3)||(Q%4!=3)||(gcd(N, R) != 1))

{

printf("\nBBS cannot be generated!!");

return 0;

}

N = P \* Q;

X = (R\*R) % N; //seeding

printf("\nEnter A(the starting value),\nB(the ending value),\nand the number of values to print:\n");

scanf("%d %d %d", &A, &B, &iter);

M = B- A +1;

for(i = 0; i < iter; i++)

{

X = (X\*X) % N;

stream[i] = X % M;

}

printf("\n");

for(i = 0; i < iter; i++)

{

printf("\n%d", stream[i] + A);

}

return 0;

}

SAMPLE OUTPUTS:

Enter A(the starting value),

B(the ending value),

and the number of values to print:

5

25

50

11

10

20

16

21

15

12

5

20

14

5

19

25

14

21

13

11

6

10

13

13

13

13

19

19

13

9

5

11

24

12

20

7

7

24

11

8

24

10

24

23

19

15

22

12

16

7

23

8

23