LAB #2

Title: Probability of Birth Child.

Initial Conditions:

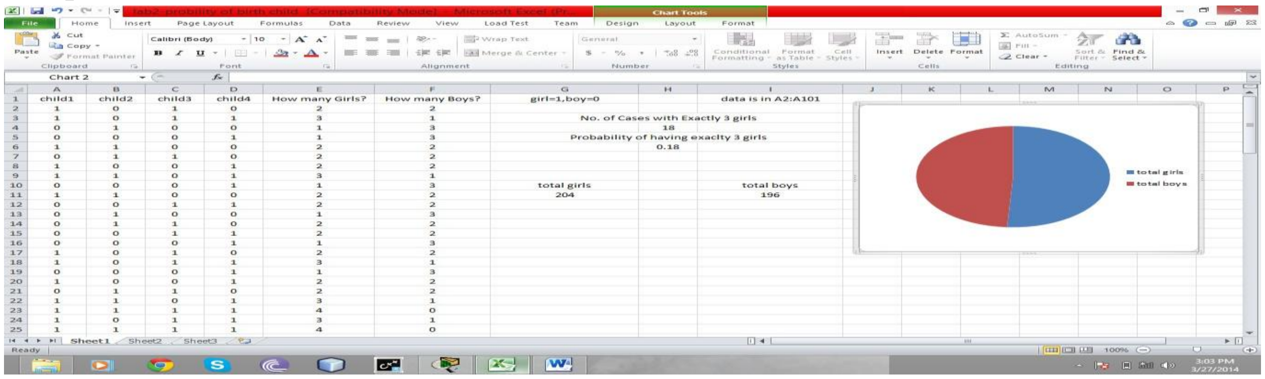
No. of children checked = 4
No. of sample cases = 100
Value assigned for girl child = 1
Value assigned for boy child = 0

Theory:
In this simulation, it is attempted to study the probability of having a boy or a girl child during a birth. The study starts with the condition that 4 births are checked at a time and the total number of sampling is done 100 times. The children are represented as child 1, child 2, child 3, and child 4foreachbirth.Then, the number of boys, girls are calculated in total and finally the probability is calculated by dividing the number by total number of children.

Procedure:

- Children are represented as child 1, child 2, child 3 and child 4 for each birth.
- If the child is girl then it is represented using 1 and if the child is boy then it is represented using 0. This is simulated by using =RANDBETWEEN (0,1) function in EXCEL which gives either 0 or 1randomly.
- Then using the =SUM(X,Y) function, the number of girls are calculated for each birth. Subtracting the number from 4 for each birth gives the number of boys.
- Then the same function is again used to calculate the total number of boys' and girls'.
- A pie chart is created using the number of boys and girls in total to display the ratio of boys and girls.
- Also number of cases with exactly 3 girls is calculated using COUNTIF ( ) function. Also the probability of having exactly 3 girls is also calculated by dividing the number by number of cases.

Sample Data:



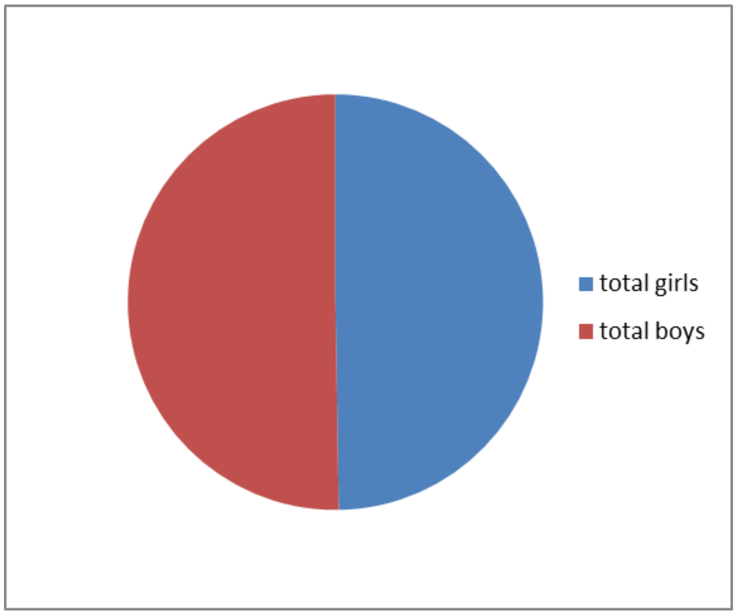
	A	B	C	D	E	F
1	child1	child2	child3	child4	How many Girls?	How many Boys?
2	1	0	1	0	2	2
3	1	0	1	1	3	1
4	0	1	0	0	1	3
5	0	0	0	1	1	3
6	1	1	0	0	2	2
7	0	1	1	0	2	2
8	1	0	0	1	2	2
9	1	1	0	1	3	1
10	0	0	0	1	1	3
11	1	1	0	0	2	2
12	0	0	1	1	2	2
13	0	1	0	0	1	3
14	0	1	1	0	2	2
15	0	0	1	1	2	2
16	0	0	0	1	1	3
17	1	0	1	0	2	2
18	1	0	1	1	3	1
19	0	0	0	1	1	3
20	1	0	0	1	2	2
21	0	1	1	0	2	2
22	1	1	0	1	3	1
23	1	1	1	1	4	0
24	1	0	1	1	3	1
25	1	1	1	1	4	0

G	H	I
girl=1,boy=0		data is in A2:A101
	No. of Cases with Exactly 3 girls	
	18	
	Probability of having exaclty 3 girls	
	0.18	
total girls		total boys
204		196

Output:

No. of Cases with Exactly 3 girls
26
Probability of having exactly 3 girls
0.26

total girls	total boys
197	199



CONCLUSION:

Hence, in this way the probability of child birth was simulated in Excel.

LAB #3

Title: To simulate the occurrences of numbers on three dices simultaneously

Initial Conditions:
We assume 3 dices with numbers 1 to 6 on them and we assume that all three dices will produce a number when thrown.

Theory:
A dice is a cube with numbers 1 to 6 marked on its six faces used in board games such as Ludo, Monopoly and so on. For the simulation, it is supposed that three dices are thrown simultaneously. The occurrence of the numbers on three dices for a throw is to be simulated. This is done by using =RANDBETWEEN (1,6) function which gives a number between 1 and 6 randomly.

Procedure:

- Three boxes or cells are taken to be the dices.
- Each box is given the function =RANDBETWEEN (1,6) to simulate them as dices.
- Whenever F9 is pressed, the values of the dices change.

Output:
Sample Case 1

4	2	1

Sample Case 2:

5	3	5

Conclusion:
Hence, in this way the simulation of occurrence of numbers on 3 dices thrown simultaneously was simulated.

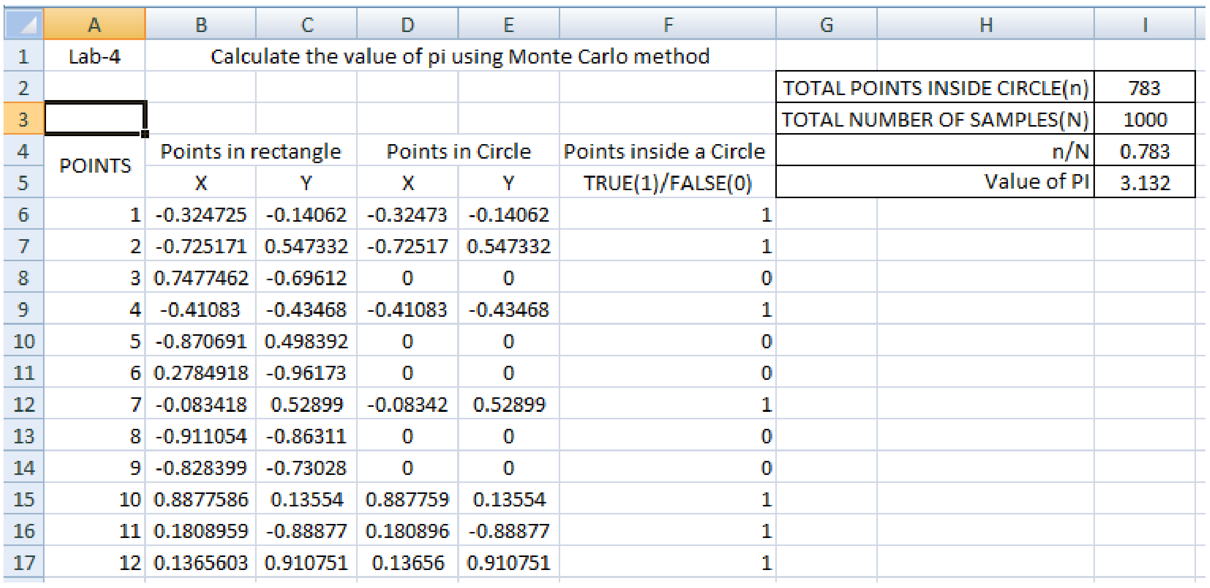
LAB #1

STATEMENT: CALCULATE THE VALUE OF PI USING MONTE CARLO

METHOD ALGORITHM:

STEP-1:	START	
STEP-2:	Plot Random points between -1 to 1 bounds on X-Axis and Y-axis for rectangle using the formula: **=1-2*RAND()**	
STEP-3:	Compute which points inside the rectangle bounds also fall under the area bounded by circle inside it, using the formula:	
	For X-axis	For Y-axis
	=IF((X^2+Y^2)<1,X,0)	**=IF((X^2+Y^2)<1,Y,0)**
STEP-4:	If the points fall inside the circle, mark those points as TRUE (1) and FALSE(0) in next column. Use the formula: **=IF((X^2+Y^2)<1,1,0)** to achieve this.	
STEP-5:	Calculate about 1000 such points for accuracy	
STEP-6:	Make a scatter diagram each for the points in Rectangle and points in Circle	
STEP-7:	Calculate the total number of points that lie inside the circle bound as found from **STEP-4**	
STEP-8:	Calculate the value of PI using the formula: **=n/N*4** Where, n = Total number of points inside circle bound N = Total number of points used	

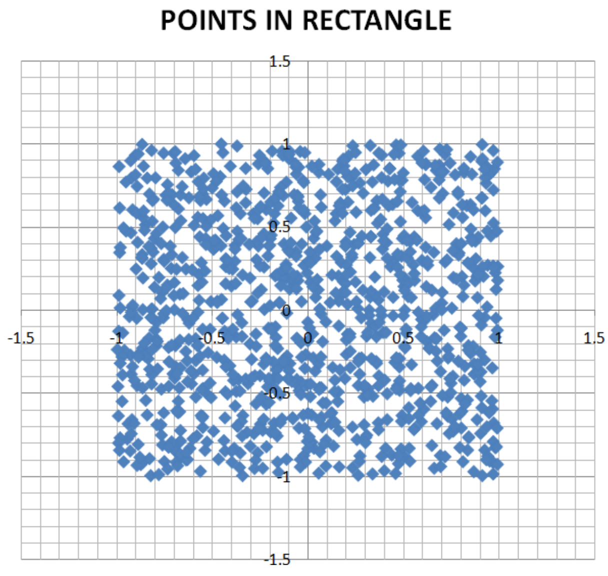
SAMPLE DATA:



	A	B	C	D	E	F	G	H	I
1	Lab-4	Calculate the value of pi using Monte Carlo method							
2							TOTAL POINTS INSIDE CIRCLE(n)		783
3							TOTAL NUMBER OF SAMPLES(N)		1000
4	POINTS	Points in rectangle		Points in Circle		Points inside a Circle		n/N	0.783
5		X	Y	X	Y	TRUE(1)/FALSE(0)		Value of PI	3.132
6	1	-0.324725	-0.14062	-0.32473	-0.14062	1			
7	2	-0.725171	0.547332	-0.72517	0.547332	1			
8	3	0.7477462	-0.69612	0	0	0			
9	4	-0.41083	-0.43468	-0.41083	-0.43468	1			
10	5	-0.870691	0.498392	0	0	0			
11	6	0.2784918	-0.96173	0	0	0			
12	7	-0.083418	0.52899	-0.08342	0.52899	1			
13	8	-0.911054	-0.86311	0	0	0			
14	9	-0.828399	-0.73028	0	0	0			
15	10	0.8877586	0.13554	0.887759	0.13554	1			
16	11	0.1808959	-0.88877	0.180896	-0.88877	1			
17	12	0.1365603	0.910751	0.13656	0.910751	1			

Fig: Sample Format for calculating value of PI using Monte Carlo method

OUTPUT:



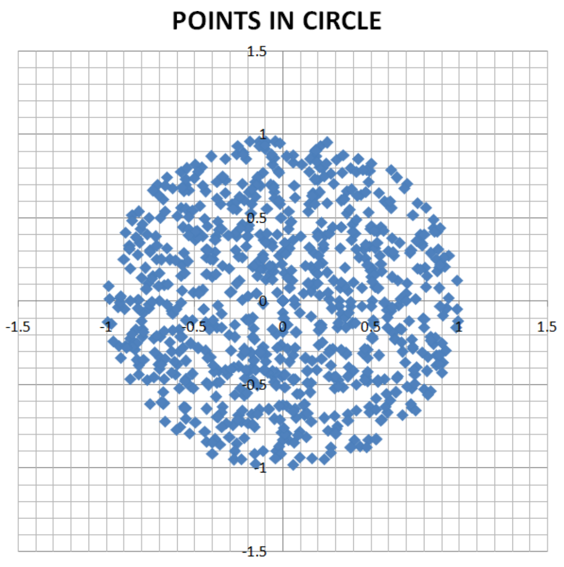


Fig-1: Plotting points of Rectangle and Circle respectively on a graph

TOTAL POINTS INSIDE CIRCLE(n)	797
TOTAL NUMBER OF SAMPLES(N)	1000
n/N	0.797
Value of PI	3.188

Table-1: Calculating the value of PI the generated points

CONCLUSION: Hence, in this way the value of Pi (π) was simulated using Monte-Carlo Method in Excel.

4.Write a computer program in C that will generate four digit random numbers using the Linear congruential method. Allow the use to input values of Xo, a, c and m.

```
#include <stdio.h>


int main() {

   int X, a, c, m;

   int count = 0;


   // User input

   printf("Enter the seed (X0): ");

   scanf("%d", &X);

   printf("Enter the multiplier (a): ");

   scanf("%d", &a);

   printf("Enter the increment (c): ");

   scanf("%d", &c);

   printf("Enter the modulus (m): ");

   scanf("%d", &m);


   printf("\nGenerating 10 four-digit random numbers using LCG:\n");


   while (count < 10) {

      X = (a * X + c) % m;

      if (X >= 1000 && X <= 9999) {

         printf("%d ", X);

         count++;

      }

   }


   printf("\n");

   return 0;

}
```

Output

```
Enter the seed (X0): 1234

Enter the multiplier (a): 1103515245

Enter the increment (c): 12345

Enter the modulus (m): 32768
```

5.Write a C program to find the weather of the next day by using Markov Chain Method.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int predictNextWeather(int currentWeather) {
   // Transition matrix
   float transition[3][3] = {
      {0.6, 0.3, 0.1},  // From Sunny
      {0.2, 0.5, 0.3},  // From Cloudy
      {0.1, 0.3, 0.6}   // From Rainy
   };

   float r = (float)rand() / RAND_MAX;
   float cumulative = 0.0;

   for (int i = 0; i < 3; i++) {
      cumulative += transition[currentWeather][i];
      if (r <= cumulative) {
         return i;
      }
   }

   return 2; // fallback
}

const char* getWeatherName(int weather) {
   switch (weather) {
      case 0: return "Sunny";
      case 1: return "Cloudy";
      case 2: return "Rainy";
```

```
        default: return "Unknown";
    }
}

int main() {
    int currentWeather;

    printf("Weather states: 0 = Sunny, 1 = Cloudy, 2 = Rainy\n");
    printf("Enter today's weather (0/1/2): ");
    scanf("%d", &currentWeather);

    // Random seed
    srand(time(NULL));

    int nextWeather = predictNextWeather(currentWeather);

    printf("Today's weather is: %s\n", getWeatherName(currentWeather));
    printf("Predicted weather for tomorrow: %s\n", getWeatherName(nextWeather));

    return 0;
}
```

Output

```
Weather states: 0 = Sunny, 1 = Cloudy, 2 = Rainy
Enter today's weather (0/1/2): 0
Today's weather is: Sunny
Predicted weather for tomorrow: Cloudy
```

6.WAP to generate random number using mid square random number generator method.

```
#include <stdio.h>
#include <math.h>

int getMiddleDigits(int num) {
   // Square the number
   long square = (long)num * num;

   // Convert to string to easily extract middle digits
   char squareStr[20];
   sprintf(squareStr, "%ld", square);

   int len = 0;
   while (squareStr[len] != '\0') len++;

   // Take the middle 4 digits (if possible)
   int start = (len - 4) / 2;
   char midStr[5];
   for (int i = 0; i < 4; i++) {
      midStr[i] = squareStr[start + i];
   }
   midStr[4] = '\0';

   return atoi(midStr);
}

int main() {
   int seed;
   printf("Enter a 4-digit seed: ");
   scanf("%d", &seed);
```

```
    if (seed < 1000 || seed > 9999) {
        printf("Please enter a valid 4-digit seed.\n");
        return 1;
    }

    printf("\nGenerating random numbers using Mid-Square Method:\n");

    for (int i = 0; i < 10; i++) {
        seed = getMiddleDigits(seed);
        printf("Random number %d: %d\n", i + 1, seed);
    }

    return 0;
}
```

OUTPUT

```
Enter a 4-digit seed: 1234


Generating random numbers using Mid-Square Method:
Random number 1: 5227
Random number 2: 3187
Random number 3: 1516
Random number 4: 2998
Random number 5: 9880
Random number 6: 6166
Random number 7: 9967
Random number 8: 9348
Random number 9: 4390
Random number 10: 2721
```

7. Write a computer program in C that will generate four digit random numbers using the multiplicative congruential method. Allow the use to input values of Xo, a, c and m.

Solution

xo=seed, x1=next random number that we will generate a=constant multiplier, c=increment, m=modulus i =for loop control, n for how many random numbers array [20] = to store the random numbers generated For Multiplicative congruential method the value of c should be zero and to generate four-digit random number the value of m should be 10000 i.e. c = 0 and m=10000.

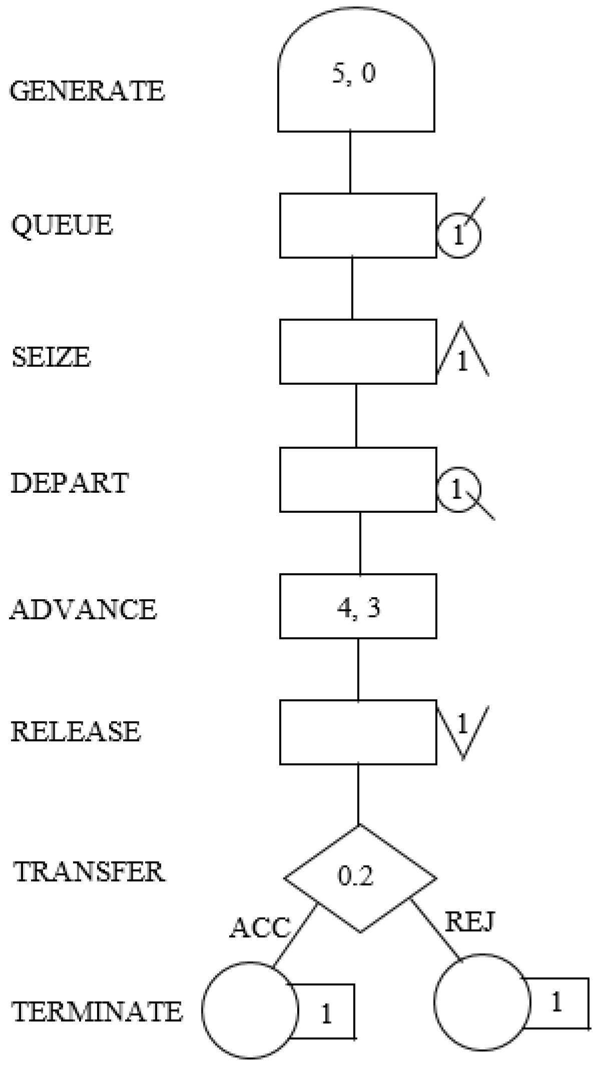
```
 #include<stdio.h>

int main() {

 int xo,x1;

int a,c,m;

 int i,n;

int array[20];

 printf("Enter the seed value xo: ");

scanf("%d",&xo);

printf("\n");

printf("Enter the constant multiplier a: ");

 scanf("%d",&a); printf("\n");

 printf("Enter the increment c: ");

scanf("%d",&c); printf("\n");

 printf("Enter the modulus m: ");

 scanf("%d",&m);

printf("\n");

 printf("How many random numbers you want to generate: ");

 scanf("%d",&n);

 printf("\n");
 for(i=0;i<n;i++) /* loop to generate random numbers */
{
 x1=(a*xo+c) %m;
 array[i]=x1;
xo=x1;
}
 printf("The generated random numbers are: ");
for(i=0;i<n;i++) {
 printf("%d",array[i]);
 printf("\t");
 }
 return(0);
 }
```

OUTPUT

```
Enter the seed value xo: 1234


Enter the constant multiplier a: 57


Enter the increment c: 10


Enter the modulus m: 10000


How many random numbers you want to generate: 4


The generated random numbers are: 348   9846    1232    234
```

8. A machine tool in a manufacturing shop is turning out parts at the rate of every 5 minutes. When they are finished, the parts are sent to an inspector, who takes 4±3 minutes to examine each one and rejects 20% of the parts. Draw and explain a block diagram for it and write a GPSS program to simulate using the concept of FACILITY.

The block diagram for given problem using GPSS is given below:



A GENERATE block is used to represent the output of the machine by creating one transaction every five units of time. A QUEUE block places the transaction in the queue and SEIZE block allows a transaction to engage a facility if it is available. If more than one inspector is available, the transaction leaves the queue which is denoted by DEPART block and enters into ADVANCE block. An ADVANCE block with a mean of 4 and modifier of 3 is used to represent inspection. The time spent on inspection will therefore be any one of the values 1, 2, 3, 4, 5, 6 or 7, with equal probability given to each value. Upon completion of the inspection, RELEASE block allows a transaction to disengage the facility and transaction go to a TRANSFER block with a selection factor of 0.2, so that 80 % of the parts go to the next location called ACC, to represent accepted parts and 20 % go to another location called REJ to represent rejects. Both locations reached from the TRANSFER block are TERMINATE blocks.

Now,

Code for simulating the given problem using GPSS:

```
    GENERATE   5, 0
    QUEUE   1
    SEIZE   1
    DEPART   1
    ADVANCE   4, 3
    RELEASE   1
    TRANSFER  0.2  ACC   REJ
ACC TERMINATE   1

REJ TERMINATE   1
```