**Lab 1: Write a program to simulate coin toss game using Monte Carlo Simulation Technique**

**Program**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main()

{

int head=0,tail=0,itr;

double r;

srand(time(NULL));

printf("Enter number of iterations\n");

scanf("%d",&itr);

for(int i=1;i<=itr;i++)

{

r =(double) rand()/RAND\_MAX;

if(r<=0.5)

head++;

else

tail++;

}

printf("Head= %d\n",head);

printf("Tail = %d\n",tail);

if(head>tail)

printf("Head wins by %d",head-tail);

else

printf("Tail wins by %d",tail-head);

}

**Lab 2 Write a C program to find out value of PI using Monte Carlo Simulation Technique**

**Program**

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

#include <string.h>

#define SEED 35791246

int main()

{

int itr=0;

double x,y;

int i,count=0;

double z;

double pi;

printf("Enter the number of iterations used to estimate pi: ");

scanf("%d",&itr);

srand(SEED);

count=0;

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

{

x = (double)rand()/RAND\_MAX;

y = (double)rand()/RAND\_MAX;

z = x\*x+y\*y;

if (z<=1.0)

count++;

}

pi=(double)count/itr\*4;

printf("Value of PI = %f",pi);

return 0;

}

**Lab 3: Write a C program to find area between given interval of a function using Monte Carlo Simulation method**

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define F(x) (x\*x)

#define xl 1

#define yl 1

#define xu 2

#define yu 4

#define N 10000

int main()

{

int xran,yran,n=0,a=1,b=2,c=4,i;

float area;

srand(time(0));

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

{

xran = (rand() % (xu - xl + 1)) + xl;

yran = (rand() % (yu - yl + 1)) + yl;

if(xran\*xran<=yran)

n++;

}

printf("Number of points = %d\n",n);

area = c\*(b-a)\*(float)n/N;

printf("The area = %f",area);

return 0;

}

**Lab 4: Write a C program to generate 10 random numbers using Linear Congruential method**

**Program:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int x0,x,a,c,M=101;

x0 = 31,a=17,c=13;

double r;

for(int i=1;i<=15;i++)

{

x = (a\*x0+c)%M;

r = (double)x/M;

printf("%f\n",r);

x0 = x;

}

}

Lab 5: Write a C program to generate 10 random numbers using mid square method

Program:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

int main()

{

long int i,s,x,y,z,nd,seed;

int n;

seed=61;

printf("How many random numbers to be generated\n");

scanf("%d",&n);

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

{

s = (seed\*seed);

y = s/10.0;

z = y/100.;

x = int((y/100.-z)\*100.);

printf("x = %d\n",x);

seed = x;

printf(" %4ld ",x);

}

}

**Lab 6: Write a C program to that tests random numbers for frequency using Kolmogorov S test**

**Program:**

#include<conio.h>

#include<stdio.h>

#define N 5

#define Dalpha 0.665

int main()

{

float R[N] = {0.05,0.14,0.44,0.81,0.93};

float X[N],Y[N],Z[N];

float D1,D2,D;

int i;

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

{

X[i-1] = (double)i/N;

}

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

{

Y[i-1] = (double)i/N-R[i-1];

}

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

{

Z[i-1] = R[i-1]-(double)(i-1)/N;

}

D1 = Y[0];

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

{

if(D1<Y[i])

D1 = Y[i];

}

D2 = Z[0];

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

{

if(D1<Z[i])

D1 = Z[i];

}

D = (D1>D2)?D1:D2;

printf("D = %0.2f",D);

if(D<Dalpha)

printf("Random numbers are uniformally distributed\n");

else

printf("Random numbers are not uniformally distributed\n");

}

**Lab 7: Write a C program that tests random numbers for frequency using chi-Square test**

**Program:**

#include<stdio.h>

#include<conio.h>

#define N 100

#define alpha 16.9

void sort(float arr[]);

int main()

{

int i,j;

float XO[10];

float XE[10];

float XOE[10];

float XOE2[10];

float R[10];

float s=0.0;

float x[] = { 0.34,0.83,0.96,0.47,0.79,0.37,0.99,0.37,0.72,0.06,0.18,0.90,

0.76,0.99,0.30,0.71,0.17,0.51,0.43,0.39,0.26,0.25,0.79,

0.77,0.17,0.23,0.99,0.54,0.56,0.84,0.97,0.89,0.64,0.67,

0.82,0.19,0.46,0.01,0.97,0.24,0.88,0.87,0.70,0.56,0.56,

0.82,0.05,0.81,0.30,0.40,0.64,0.44,0.81,0.41,0.05,0.93,

0.66,0.028,0.94,0.64,0.47,0.12,0.94,0.52,0.45,0.65,0.10,

0.69,0.96,0.40,0.60,0.21,0.74,0.73,0.31,0.37,0.42,0.34,

0.58,0.19,0.11,0.46,0.22,0.99,0.78,0.39,0.18,0.75,0.73,0.79,

0.29,0.67,0.74,0.02,0.05,0.42,0.49,0.49,0.05,0.62,0.78 };

sort(x);

for(i=0;i<10;i++)

{

XO[i]=0.0;

XE[i]=10.0;

}

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

{

if(x[i]<=0.1)

XO[0]++;

else if(x[i]<=0.2)

XO[1]++;

else if(x[i]<=0.3)

XO[2]++;

else if(x[i]<=0.4)

XO[3]++;

else if(x[i]<=0.5)

XO[4]++;

else if(x[i]<=0.6)

XO[5]++;

else if(x[i]<=0.7)

XO[6]++;

else if(x[i]<=0.8)

XO[7]++;

else if(x[i]<=0.9)

XO[8]++;

else if(x[i]<=1.0)

XO[9]++;

}

for(i=0;i<10;i++)

{

XOE[i] = XO[i]-XE[i];

XOE2[i] = XOE[i]\*XOE[i];

R[i] = XOE2[i]/XE[i];

s = s+R[i];

}

printf("s = %0.2f\n",s);

printf("Alpha at 5%% level of significance for n=9 is %0.2f\n",alpha);

if(s<=alpha)

printf("Accepted");

else

printf("Rejected");

}

void sort(float x[])

{

int i,j;

float temp;

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

{

for(j=0;j<N-1;j++)

{

if(x[j+1]<x[j])

{

temp = x[j];

x[j] = x[j+1];

x[j+1] = temp;

}

}

}

}

**Lab 8: Write a program to test random numbers for independence using autocorrelation method**

**Programs:**

#define N 30

#include<stdio.h>

#include<math.h>

int main()

{

int i,m,M,k;

i=2,m=5;

M = ((N-i)/m)-1;

float s35=0,r35,z0;

float R[] = {0.12,0.01,0.23,0.28,0.89,0.31,0.64,0.28,0.83,0.93,0.99,

0.15,0.33,0.35,0.91,0.41,0.60,0.27,0.75,0.88,0.68,0.49,0.05,0.43,0.95,

0.58,0.19,0.36,0.69,0.87};

for(k=0;k<=M;k++)

{

s35 =s35+R[i+k\*m]\*R[i+(k+1)\*m];

}

s35 = s35/(M+1);

s35 = s35-0.25;

r35 = sqrt(13\*M+7)/(12\*(M+1));

z0 = r35/s35;

if(z0<=1.96)

printf("The null hypothesis that numbers are independent is accepted\n");

else

printf("The null hypothesis that numbers are independent is not accepted\n");

return 0;

}

**Lab 9: Write a program to test whether the given matrix is Markov or not**

**Program**

#include<stdio.h>

#define N 3

#define M 3

int isMarkovMatrix(float m[][N])

{

int i,j,s,t=1;

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

{

s=0;

for(j=0;j<N;j++)

{

s = s+m[i][j];

}

if(s>1)

{

t=0;

break;

}

}

return t;

}

void read(float m[][N])

{

printf("Enter element of %d\*%d matrix\n",M,N);

for(int i=0;i<M;i++)

{

for(int j=0;j<N;j++)

{

scanf("%f",&m[i][j]);

}

}

}

int main()

{

float matrix[M][N];

read(matrix);

if(isMarkovMatrix(matrix))

printf("The matrix is Markov Matrix\n");

else

printf("The matrix is not Markov matrix\n");

return 0;

}

**Lab 10: Write a program to simulate the game called DiceToss.**

**Program:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<time.h>

int Arand()

{

int r;

r = 1+rand()%5;

return r;

}

int Brand()

{

int r;

r = 1+rand()%5;

return r;

}

int main()

{

srand(time(0));

int sa=0,sb,a,b,x,y;

a = Arand();

b = Arand();

sa = a+b;

x = Brand();

y = Brand();

sb = x+y;

if(sa>sb)

printf("A wins the game by %d points\n",sa);

else

printf("B wins the game by %d points\n",sb);

return 0;

}

**Program 11: Write C program to simulate single server queuing system**

**Program:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

int main()

{

int kk,i,j,run=10;

float x,iat,st, awt, pcu,wt=0,it=0;

float mean=10.0, sd= 1.5, mue=9.5, sigma=1.0;

float sb = 0.0,se=0.0,cit=0, cat=0, cwt=0;

printf("\nIAT CAT SB ST SE CWT CIT");

for(j=1;j<=run;++j)

{

float sum=0;

for(i=1;i<=12;++i)

{

x = rand()/32768.0;

sum = sum+x;

}

iat = mean+sd\*(sum-6.0);

cat = cat+iat;

if(cat<=se)

{

sb = se;

wt = se-cat;

cwt = cwt+wt;

}

else

{

sb = cat;

it = sb-se;

cit = cit+it;

}

sum = 0;

for(i=1;i<=12;++i)

{

x = rand()/32768.0;

sum = sum+x;

}

st = mue+sigma\*(sum-6.0);

se = sb+st;

printf("\n %5.2f %6.2f %6.2f %6.2f %6.2f %6.2f %6.2f",iat,cat,sb,st, se, cwt,cit);

}

awt = cwt/run;

pcu = ((cat-cit)\*100.0)/cat;

printf("\n Average waiting time = %6.2f",awt);

printf("\nPercentage capacity untilization = %6.2f",pcu);

}

**Lab 12 : Write a GPSS to simulate following problem**

A machine tool in a manufacturing shop is turning out parts at the rate of one every 5 minutes. As they are finished, the parts go to an inspector, who take 4+-3 minutes to examine each one and rejects about 10 % of the parts. Each part will be represented by one transaction and the time unit selected for the problem will be 1 minute

Program:

GENERATE 5

ADVANCE 4,3

TRANSFER 0.1, ACC,REJ

ACC TERMINATE 1

REJ TERMINATE 1

START 1000

**Lab 13:** Write a program in GPSS to simulate barber shop problem

Program:

GENERATE 18,6

ADVANCE 1

QUEUE SEAT

SEIZE JOE

DEPART SEAT

ADVANCE 15,3

RELEASE JOE

TERMINATE 0

GENERATE 540

TERMINATE 1

START 1