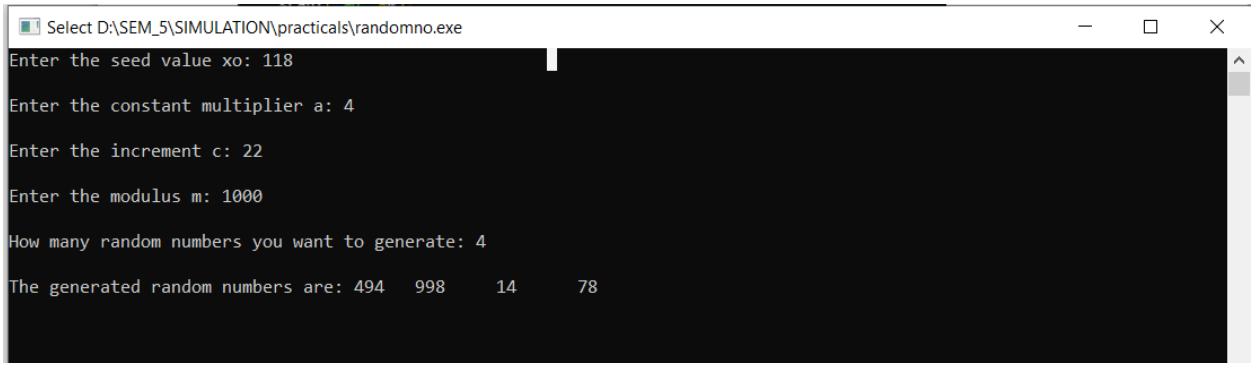
INDEX

Problems	Remarks
1. Simulate Linear Congruential Method for Random Number Generation	
2. Simulate KS Test	
3. Calculate the value of PI using Monte Carlo Method	
4. Simulate Barber Shop in GPSS	
5. Simulate M-M-1 Queuing System	

C Program of Linear Congruential Method

```
#include<stdio.h>
#include<conio.h>
int main()
{
	int xo,x1; /*xo=seed, x1=next random number that we will generate */
	int a,c,m; /*a=constant multiplier, c=increment, m=modulus */
	int i,n; /*i for loopcontrol, n for how many random numbers */
	int array[20]; /*to store the random numbers generated */
	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");
    }
    getch();
    return(0);
}
```



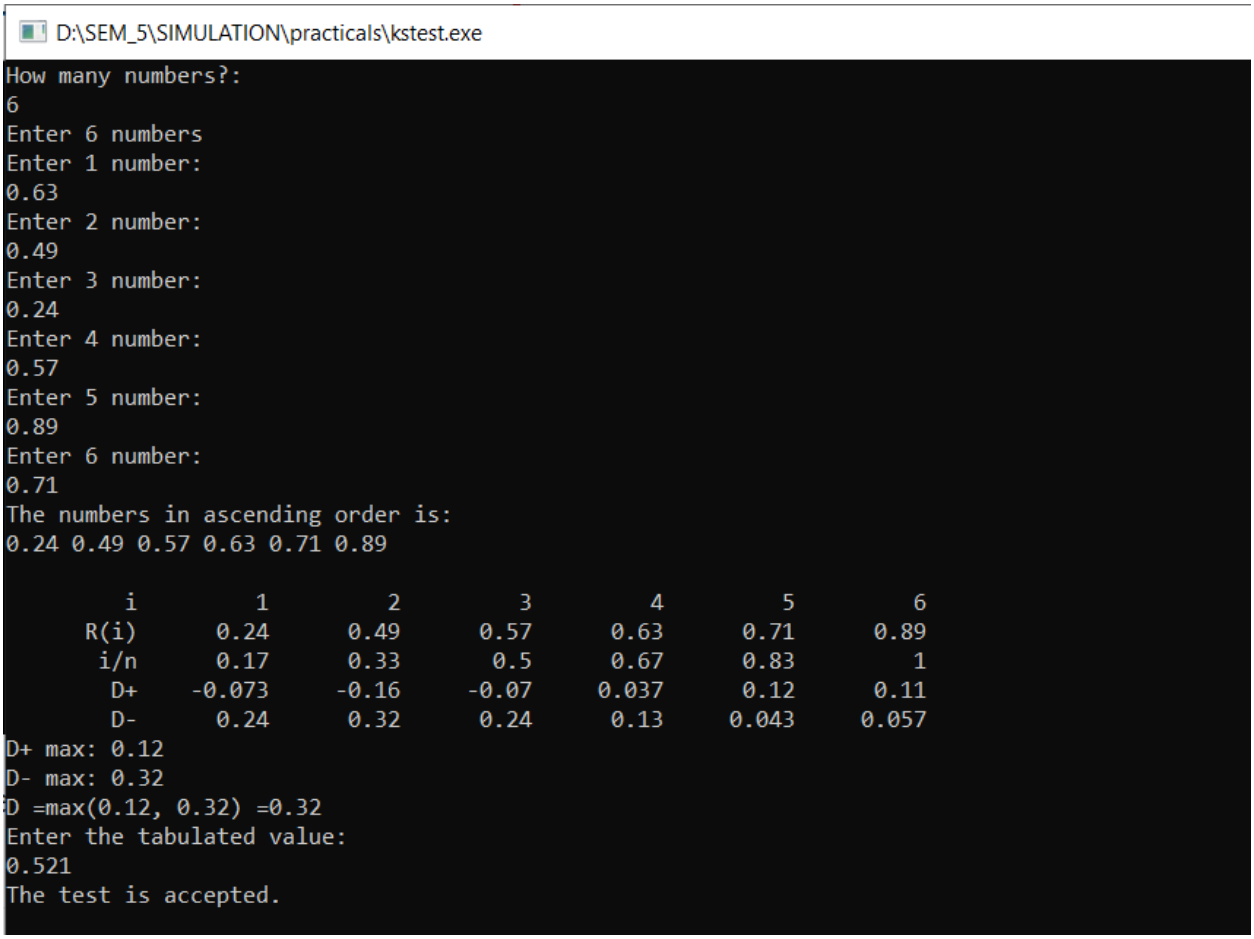
C++ Program of Kolmogorov Smirnovks (KS) Test

```
#include<iostream>
#include<conio.h>
#include<iomanip>
using namespace std;
class KS
{
private:
float numbers[20];
float D,tabulatedD;
float Dplusmax,Dminusmax;
float Dplus[20],Dminus[20];
float ratio[20],ratiominus[20];
int i,j,n;
public:
void getdata() //to get the random numbers
{
cout<<"How many numbers?:"<<endl;
cin>>n;
cout<<"Enter "<<n<<" numbers"<<endl;
for(i=0;i<n;i++)
{
cout<<"Enter "<<i+1<<" number:"<<endl;
cin>>numbers[i];
}
}
float BubbleSort() // arrange the number in increasing order
{
int i,j;
float temp;
for(i=0;i<n-1;i++)
{
for(j=0;j<n-i-1;j++)
{
if(numbers[j]>numbers[j+1])
{
temp=numbers[j];
numbers[j]=numbers[j+1];
numbers[j+1]=temp;
}
}
}
```

```
cout<<"The numbers in ascending order is:"<<endl;
for(i=0;i<n;i++)
{
cout<<setprecision(2)<<numbers[i]<<" ";
}
}
void calculate() // find D+, D-
{
for(i=0;i<n;i++)
{
int j;
j=i+1;
ratio[i]=(float)j/n;
ratiominus[i]=(float)i/n;
Dplus[i]=ratio[i]-numbers[i];
Dminus[i]=numbers[i]-ratiominus[i];
}
}
void display() // display the tabulated format and find D
{
cout<<endl;
cout<<endl;
cout<<setw(10)<<"i";
for(i=1;i<=n;i++)
{
cout<<setw(10)<<i;
}
cout<<endl;
cout<<setw(10)<<"R(i)";
for(i=0;i<n;i++)
{
cout<<setw(10)<<numbers[i];
}
cout<<endl;
cout<<setw(10)<<"i/n";

for(i=0;i<n;i++)
{
cout<<setw(10)<<setprecision(2)<<ratio[i];
}
cout<<endl;
cout<<setw(10)<<"D+";
for(i=0;i<n;i++)
{
cout<<setw(10)<<setprecision(2)<<Dplus[i];
}
cout<<endl;
```

```
cout<<setw(10)<<"D-";
for(i=0;i<n;i++)
{
cout<<setw(10)<<setprecision(2)<<D
minus[i];
}
cout<<endl;
Dplusmax=Dplus[0];
Dminusmax=Dminus[0];
for(i=1;i<n;i++)
{
if(Dplus[i]>Dplusmax)
{
Dplusmax=Dplus[i];
}
if(Dminus[i]>Dminusmax)
{
Dminusmax=Dminus[i];
}
}
cout<<"D+ max:
"<<Dplusmax<<endl;
cout<<"D- max:
"<<Dminusmax<<endl;
cout<<"D =max("<<Dplusmax<<",
"<<Dminusmax<<") =";
if(Dplusmax>Dminusmax)
{
D=Dplusmax;
}
else
{
D=Dminusmax;
}
cout<<D;
cout<<endl;
}
void conclusion() // asking tabulated
D and comparing it with
D(calculated)
{
cout<<"Enter the tabulated
value:"<<endl;
cin>>tabulatedD;
if(D<tabulatedD)
{
cout<<"The test is accepted."<<endl;
}
else
{
cout<<"The test is rejected."<<endl;
}
}
```

```
};

int main() //main function
{
KS ks1; //object of KS class
ks1.getdata(); //function calls
ks1.BubbleSort();
ks1.calculate();
ks1.display();
ks1.conclusion();
getch();
return(0);
}
```

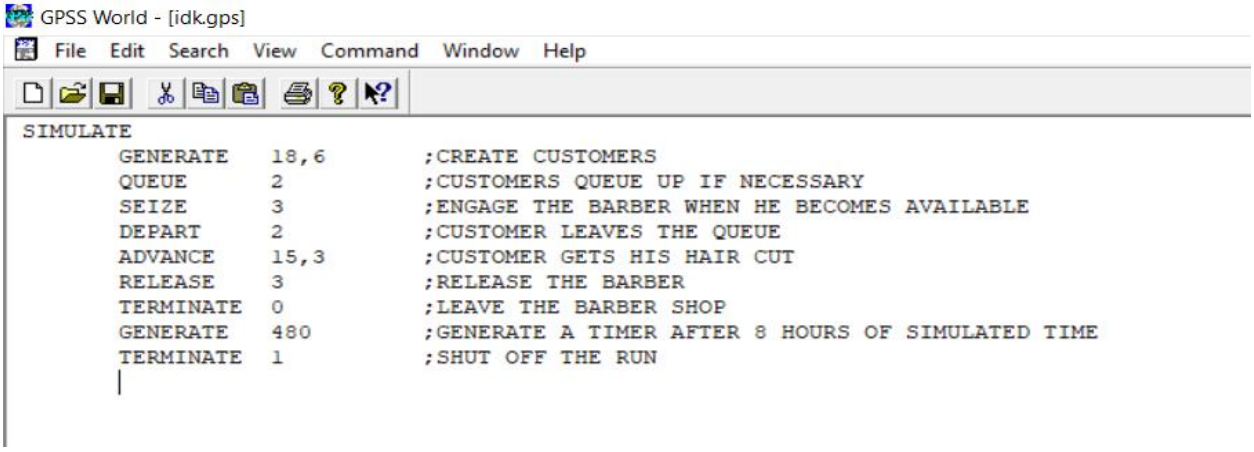


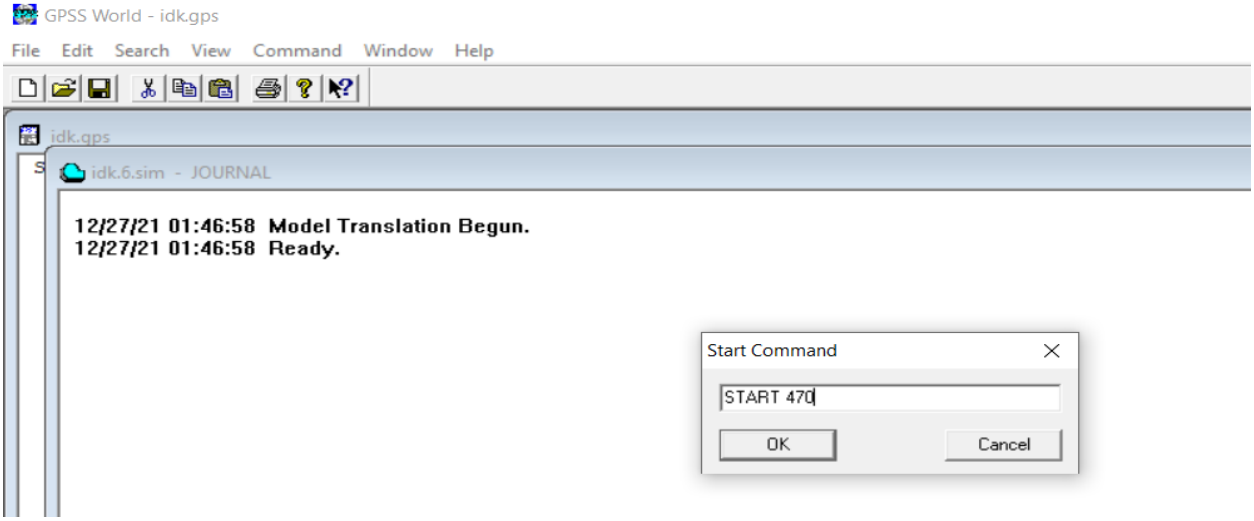
Estimate the value of PI using Monte Carlo Method

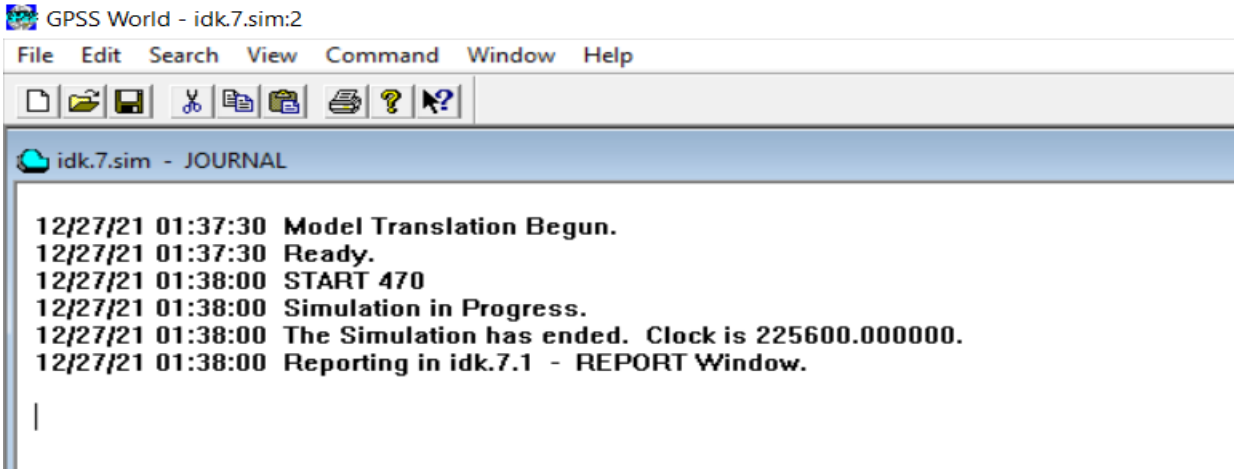
```
import random
INTERVAL= 1000
circle_points= 0
square_points= 0

# Total Random numbers generated= possible x
# values* possible y values
for i in range(INTERVAL**2):

    # Randomly generated x and y values from a
    # uniform distribution
    # Range of x and y values is -1 to 1
    rand_x= random.uniform(-1, 1)
    rand_y= random.uniform(-1, 1)

    # Distance between (x, y) from the origin
    origin_dist= rand_x**2 + rand_y**2

    # Checking if (x, y) lies inside the circle
    if origin_dist<= 1:
        circle_points+= 1

    square_points+= 1

    # Estimating value of pi,
    # pi= 4*(no. of points generated inside the
    # circle)/ (no. of points generated inside the square)
    pi = 4* circle_points/ square_points

## print(rand_x, rand_y, circle_points, square_points, "-", pi)
## print("\n")

print("Final Estimation of Pi=", pi)
```

```
PS C:\xampp\htdocs\RANJAN> & C:/Users/Asus/AppData/Local/Microsoft/WindowsApps/python3.9.exe d:/SEM_5/SIMULATION/practicals
/pi.py
Final Estimation of Pi= 3.140708
```

GPSS Barber Shop Simulation







GPSS World - [idk.7.1 - REPORT]

File Edit Search View Command Window Help

```
              GPSS World Simulation Report - idk.7.1


                   Monday, December 27, 2021 01:38:00

           START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
                0.000         225600.000     9         1          0


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE         12526              0       0
                    2    QUEUE            12526              0       0
                    3    SEIZE            12526              0       0
                    4    DEPART           12526              0       0
                    5    ADVANCE          12526              1       0
                    6    RELEASE          12525              0       0
                    7    TERMINATE        12525              0       0
                    8    GENERATE           470              0       0
                    9    TERMINATE          470              0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 3                 12526    0.831      14.962  1    12996    0    0     0      0


QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 2                    2    0  12526   8601     0.046      0.828      2.643   0


FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
 12996     0       225610.223   12996      5      6
 12997     0       225612.209   12997      0      1
 12998     0       226080.000   12998      0      8
```

GPSS World - [idk.7.sim:3 - BLOCK ENTITIES]

File Edit Search View Command Window Help

Location | Find | Continue | Halt | Step | Place | Remove

Loc	Block Type	Current Count	Entry Count	Retry Chain	Line Number	Include-file
1 GEN	GENERATE	0	12526	0	2	0
2 QUE	QUEUE	0	12526	0	3	0
3 SEI	SEIZE	0	12526	0	4	0
4 DEP	DEPART	0	12526	0	5	0
5 ADV	ADVANCE	1	12526	0	6	0
6 REL	RELEASE	0	12525	0	7	0
7 TER	TERMINATE	0	12525	0	8	0
8 GEN	GENERATE	0	470	0	9	0
9 TER	TERMINATE	0	470	0	10	0

M-M-1 Queue Simulation

```
import numpy as np
import queue
import copy
import matplotlib.pyplot as plt
# Input Parameters
total_time = int(input("Enter time
for simulation (Hours): "))
IAT_rate = int(input("Enter Job
Arrival Rate (/Hour): "))
ST_rate = int(input("Enter Job
Service Rate (/Hour): "))
rho = IAT_rate/ST_rate
# Initialize Parameters
qu = queue.Queue()
curr_process = None
IAT = []
ST = []
AT = []
wait_time = []
server_busy = False
list_wait = []
list_delay = []
num_processes =
int(np.random.poisson(IAT_rate)*
total_time)
num_processes_served = 0
# Populate Inter-Arrival-Times (IAT)
for i in range(num_processes):
    temp =
np.random.exponential(1/IAT_rate)*60*
60
    if i==0:
        IAT.append(0)
    else:
        IAT.append(int(temp -
temp%1))
# Populate Service-Times (ST) (where
ST[i]!=0)
while not len(ST) == num_processes:
    temp =
np.random.exponential(1/ST_rate)*60*6
0
    if not int(temp- temp%1)<1:
        ST.append(int(temp - temp%1))
# Save a copy of ST
ST_copy = copy.deepcopy(ST)
# Get Arrival-Times (AT) from IAT
starting at t=0
# and initialize Waiting-Times to 0
for i in range(num_processes):
    if i == 0:
        AT.append(0)
    else:
        AT.append(AT[i-1] + IAT[i])
    wait_time.append(0)
# Simulation of M/M/1 Queue (i
represents current time)
for i in range(total_time*60*60):
    if server_busy:
        for item in list(qu.queue):
            wait_time[item] =
wait_time[item] + 1
        ST[curr_process] =
ST[curr_process] - 1
        if ST[curr_process] == 0:
            server_busy = False
            num_processes_served =
num_processes_served + 1
    for j in range(num_processes):
        if i== AT[j]:
            qu.put(j)
    if not server_busy and not
qu.empty():
        curr_process = qu.get()
        server_busy = True
    sum_wait = 0
    sum_delay = 0
    for i in
range(num_processes_served):
        sum_wait = sum_wait +
wait_time[i]
        sum_delay = sum_delay +
wait_time[i] + ST_copy[i]
    if num_processes_served == 0:
```

```
        list_wait.append(0)
        list_delay.append(0)
    else:
        list_wait.append(sum_wait/(nu
m_processes_served*60*60))
        list_delay.append(sum_delay/(
num_processes_served*60*60))
```

```
plt.plot([i+1 for i in
range(total_time*60*60)], list_wait)
plt.ylabel("Avg Wait Times")
plt.show()
plt.plot([i+1 for i in
range(total_time*60*60)], list_delay)
plt.ylabel("Avg Delay Times")
plt.show()
```

```
PS C:\xampp\htdocs\RANJAN> & C:/Users/Asus/AppData/Local/Microsoft/WindowsApps/python3.9.exe c:/xampp/htdocs/RANJAN/lab/mm1
.py
Enter time for simulation (Hours): 1
Enter Job Arrival Rate (/Hour): 60
Enter Job Service Rate (/Hour): 75
```