**Lab-8**

## Aim: Implement a Single Server Queuing System.

### Code:

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

double getRandom() {

return (double(rand())/RAND\_MAX);

}

class ExponentialDistribution {

double mu;

public:

ExponentialDistribution(double m) {mu=m;}

double generateRandomVariate() {

return (-1/mu)\*log(getRandom());

}

};

int main() {

double meanArrival, meanService, nextArrivalTime=0, totalIdleTime=0, idleTime, totalWaitTime=0, waitTime, nextDepartureTime=0, nextServiceBeginTime, service, totalMinutes;

int requestsServed=0;

cout<<"Enter Mean Arrival Rate (per hour): "; cin>>meanArrival;

cout<<"Enter Mean Service Rate (per hour): "; cin>>meanService;

cout<<"Enter Total Simulation Hours: "; cin>>totalMinutes;

totalMinutes=totalMinutes\*60;

ExponentialDistribution interArrivalTime(meanArrival/60), serviceTime(meanService/60);

cout<<"R.No.\tArrival\_Time\tService\_Begin\tService\_Time\tDeparture\tWait\_Time\tIdle\_Time"<<endl;

while(nextDepartureTime<=totalMinutes)

{

nextArrivalTime+=interArrivalTime.generateRandomVariate();

if(nextArrivalTime<=nextDepartureTime)

{

nextServiceBeginTime=nextDepartureTime;

waitTime=nextDepartureTime-nextArrivalTime;

totalWaitTime+=waitTime;

idleTime=0;

}

else

{

nextServiceBeginTime=nextArrivalTime;

idleTime=nextArrivalTime-nextDepartureTime;

totalIdleTime+=idleTime;

waitTime=0;

}

service=serviceTime.generateRandomVariate();

nextDepartureTime=nextServiceBeginTime+service;

++requestsServed;

cout<<setprecision(5)<<requestsServed<<"\t"<<nextArrivalTime<<"\t\t"<<nextServiceBeginTime<<"\t\t"<<service<<"\t\t"<<nextDepartureTime<<"\t\t"<<waitTime<<"\t\t"<<idleTime<<endl;

}

cout<<"Average Wait Time: "<<totalWaitTime/requestsServed<<endl;

cout<<"Idle Time Percentage: "<<totalIdleTime/totalMinutes\*100<<endl;

cout<<"Capacity Utilization: "<<(nextArrivalTime-totalIdleTime)/nextArrivalTime\*100<<endl;

return 0;

}

### output:

