

Bank system simulation

Data Structures Project





|  |  |
| --- | --- |
| Names | section |
| Fady fares abdel Ahad sobhy | 2 |
| Farida ali mohamed | 2 |
| Reem medhat arnest | 2 |

Names: Fady fares

Table of Contents

[2 Analysis 3](#_Toc417943298)

[2.1 Problem statement: 3](#_Toc417943299)

[2.2 problem breakdown: 3](#_Toc417943300)

[2.3 use case diagrams 4](#_Toc417943301)

[2.4 use case description: 4](#_Toc417943302)

[2.4.1 U1: Add Clerk 4](#_Toc417943303)

[2.4.2 U2: Add Customer 5](#_Toc417943304)

[2.4.3 U3: Simulate 5](#_Toc417943305)

[2.4.4 U4: Enter next customer 5](#_Toc417943306)

[3 Assumptions 6](#_Toc417943307)

[3.1 Clerk 6](#_Toc417943308)

[3.2 States 6](#_Toc417943309)

[3.3 turn 6](#_Toc417943310)

[3.4 Time 6](#_Toc417943311)

[3.5 ticket 6](#_Toc417943312)

[4 Design 7](#_Toc417943313)

[4.1 Class Diagram 7](#_Toc417943314)

[4.2 Activity Diagram 8](#_Toc417943315)

[5 user Manual 10](#_Toc417943316)

[5.1 welcome screen 10](#_Toc417943317)

[5.2 Choose input type 10](#_Toc417943318)

[5.2.1 From file 10](#_Toc417943319)

[5.2.2 From console 12](#_Toc417943320)

[5.3 output files 13](#_Toc417943321)

[5.3.1 Typical Output of the file input 13](#_Toc417943322)

[5.4 Simulation 15](#_Toc417943323)

[5.4.1 Typical simulation for input from file deposit service 16](#_Toc417943324)

[6 The code 22](#_Toc417943325)

[6.1 node.h 22](#_Toc417943326)

[6.2 Linked list.h 22](#_Toc417943327)

[6.3 customer.h 23](#_Toc417943328)

[6.4 queue.h 24](#_Toc417943329)

[6.5 clerk.h 24](#_Toc417943330)

[6.6 main .cpp 25](#_Toc417943331)

[6.6.1 Initialization and global variables 25](#_Toc417943332)

[6.6.2 int main() 26](#_Toc417943333)

[6.6.3 void ExtraClerk() 26](#_Toc417943334)

[6.6.4 void enterCustomer () 27](#_Toc417943335)

[6.6.5 void selectSimulate() 29](#_Toc417943336)

[6.6.6 void chooseInput() 30](#_Toc417943337)

[6.6.7 void printData(clerk &c) 30](#_Toc417943338)

[6.6.8 void setCustomerData(clerk &c) 31](#_Toc417943339)

[6.6.9 template<typename T> void printElement(T t,int width) 31](#_Toc417943340)

[6.6.10 void readFile(string fileName) 32](#_Toc417943341)

[6.6.11 void writeToFile(string filename,clerk &c) 34](#_Toc417943342)

[6.6.12 void SetData\_to\_file(clerk&c) 35](#_Toc417943343)

[6.6.13 void draw\_Tobank() 35](#_Toc417943344)

[6.6.14 void draw\_Simulation() 35](#_Toc417943345)

[6.6.15 void draw\_welcome() 36](#_Toc417943346)

[6.6.16 void max\_finish(int x) 36](#_Toc417943347)

[6.6.17 void statistics(string filename) 36](#_Toc417943348)

# Analysis

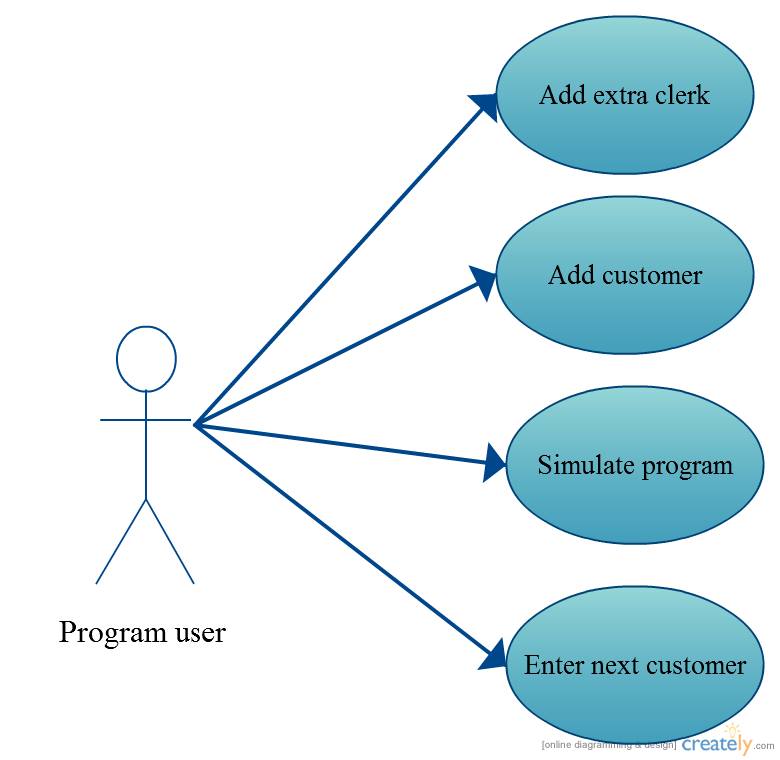
## Problem statement:

“Use queue and linked list data structure to simulate a Bank system where customers enter the system to be served by bank clerks. Each bank clerk needs some time to finish serving the customer. Customer will enter the bank to withdraw, depose or transfer money between different accounts. Each of these services needs expected time. Your program user will decide when to enter new customers and which type and also decide the current time; your program should display the current state of the waiting and served, and expected time to finish all customers. Add some option to add new clerks and create a ticket displayed to each new coming customers telling her/him how many customers are waiting before her/him.”

## problem breakdown:

1. Customer enter the system to be served by the bank clerk.
2. Each bank clerk needs some time to finish serving the customer.
3. Customer will enter the bank to withdraw, depose, or transfer money between different accounts.
4. The user will decide when to enter new customer.
5. The program will display the current state of the customer.
6. New clerk can be added to the system.

## use case diagrams



## use case description:

### U1: Add Clerk

Precondition: added clerks don’t exceed 2 for each service (number of clerks is 1 initially).

1. User enters ‘y’ if he wants to add second clerk for a certain service, otherwise he should enter ‘n’.
2. Post condition: a clerk is added to the service list.

Abnormal paths: If the user tried to enter any other character rather than ‘y’ or ‘n’ it will be neglected.

### U2: Add Customer

1. User decides which service he wants to add a customer to.
2. User enters ‘q’ when he finishes adding customers.

Post condition: Customer is added in a proper queue of certain service.

Abnormal path: If the user entered any other character rather than any of the services or ‘q’, this character will be neglected.

### U3: Simulate

Precondition: user should at least add one customer in a certain service.

1. User choose which service to simulate.
2. User can specify a certain instance to simulate for all services as input in the file.
3. After simulation finishes for a certain service, user can enter ‘q’ to finish simulation.

Abnormal path: If the user entered any other character rather than any of the services or ‘q’, this character will be neglected.

### U4: Enter next customer

Precondition: user should at least add one customer in a certain service.

1. User enters any key to let the next customer to be served and states and the turns of customers will be updated till all customers are served.

Abnormal path: If the user pressed “SPACE” or “ENTER” ,it will be neglected until any character is entered.

# Assumptions

## Clerk

* Initially we have 3 clerks each one will do certain function (withdraw, deposit, and transfer money).
* Max number of clerk on each service is 2.
* The clerk will have to press any key to continue the work and serves the next customer in the queue.

## States

## turn

1. Any +ve number will indicates to your order in the queue or in other words the number of customers before you in the turn to the clerk, and your state is “waiting”.
2. 0 means that it’s your turn and your state is “serving”.
3. -1 means that you have finished your service successfully and your state is “served”.

## Time

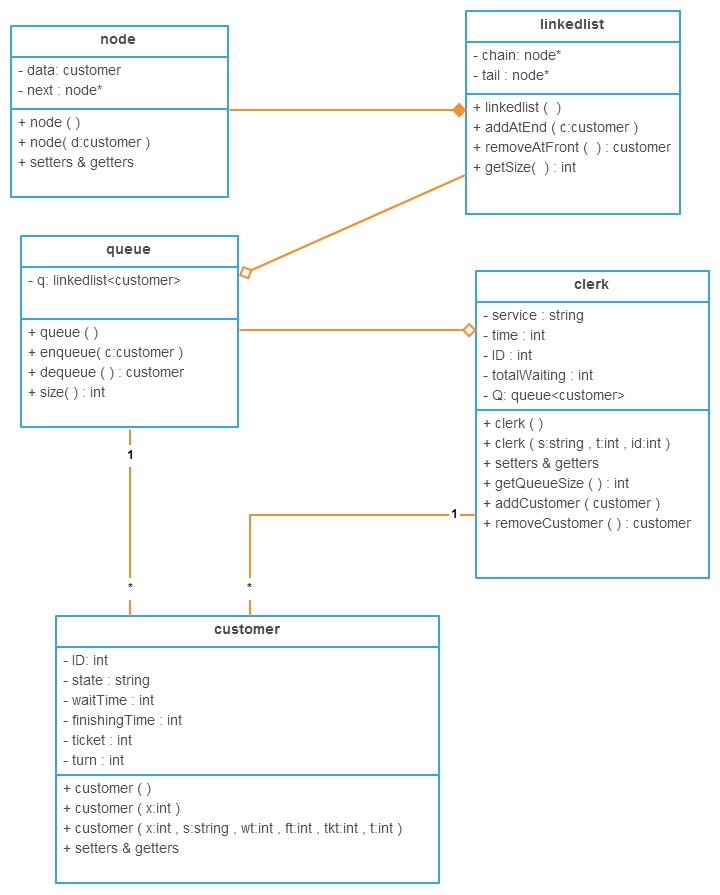
* Time for withdraw is estimated to be 10 minutes.
* Time for deposit is estimated to be 5 minutes.
* Time for transfer money is estimated to be 7 minutes.

## ticket

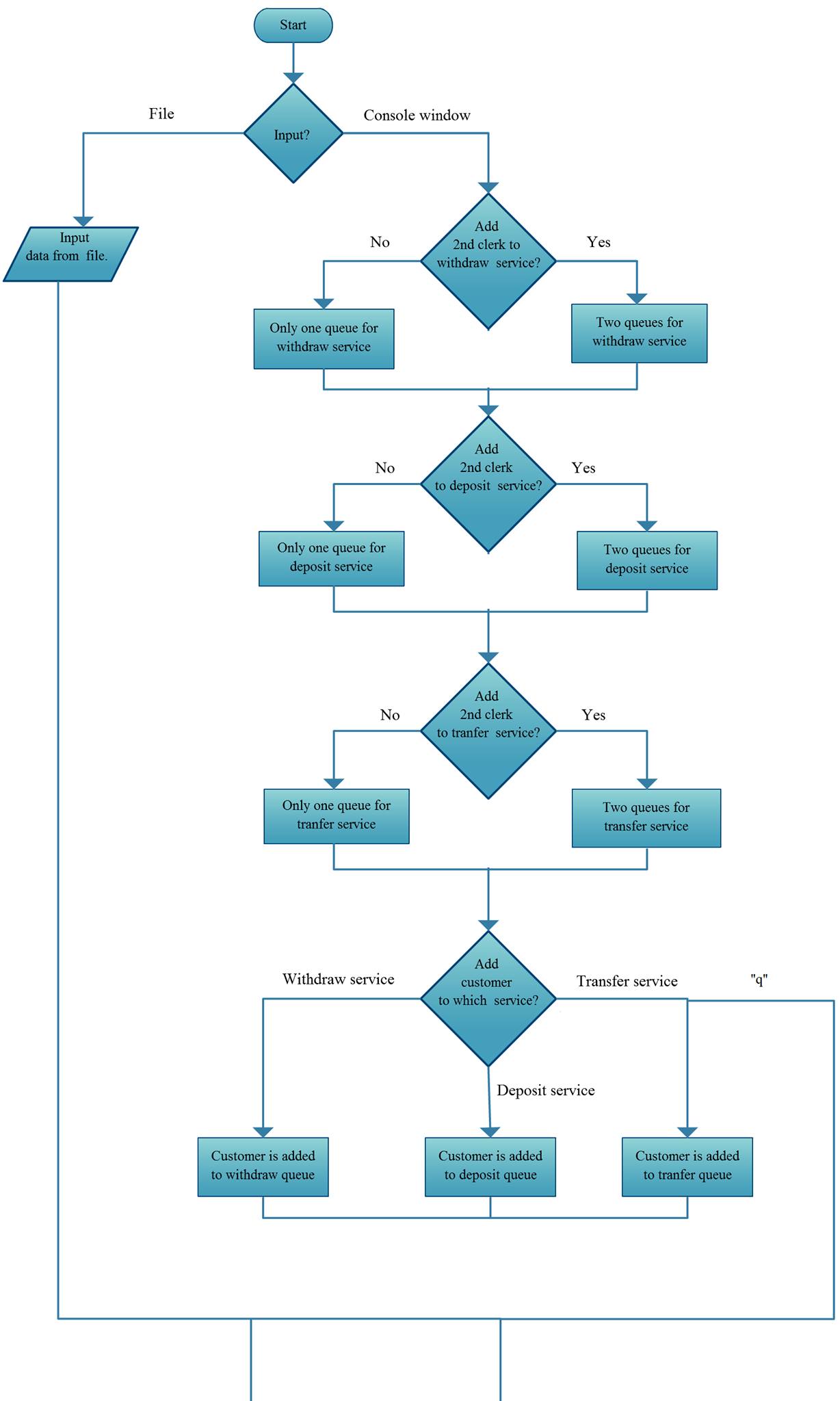
1. It’s your initial turn when you first entered the queue of a certain service.

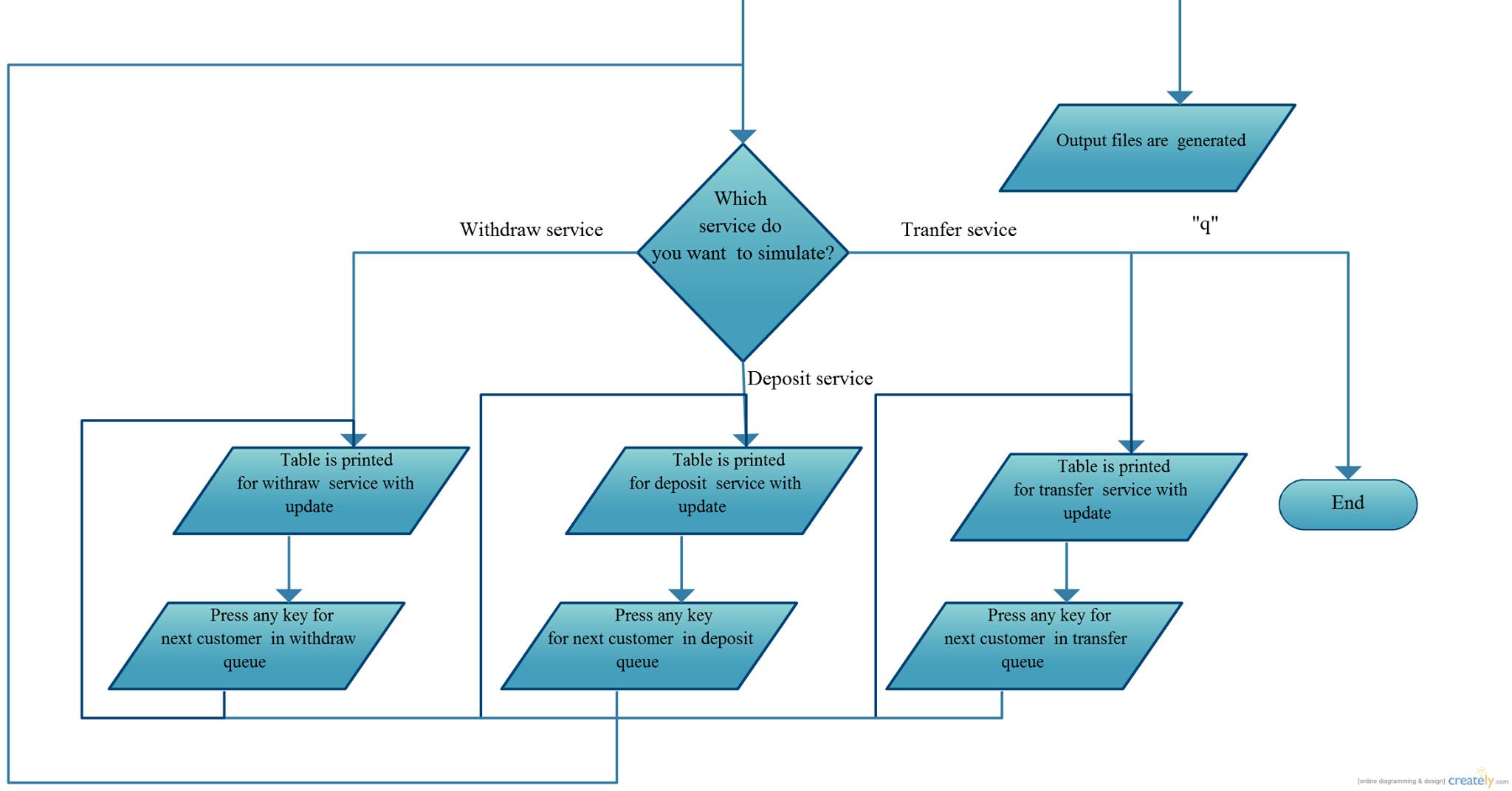
# Design

## Class Diagram



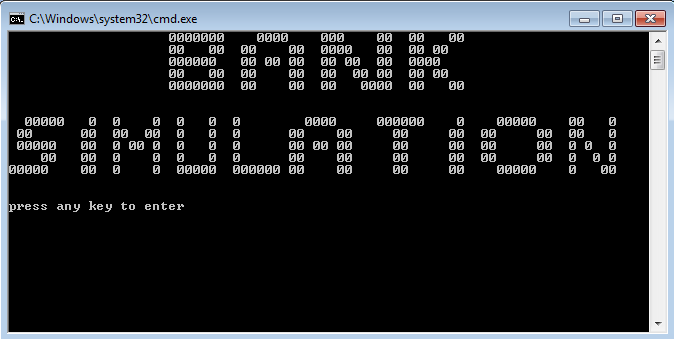
## Activity Diagram





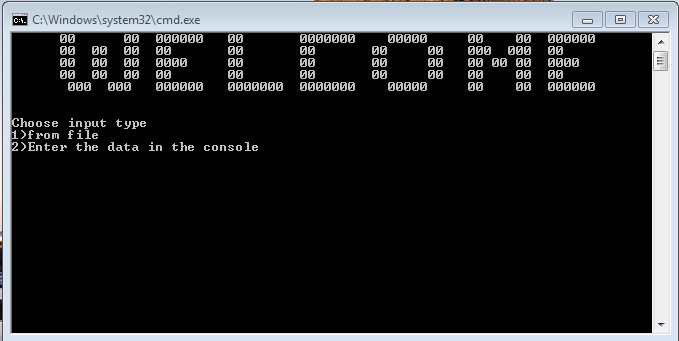
# user Manual

## welcome screen



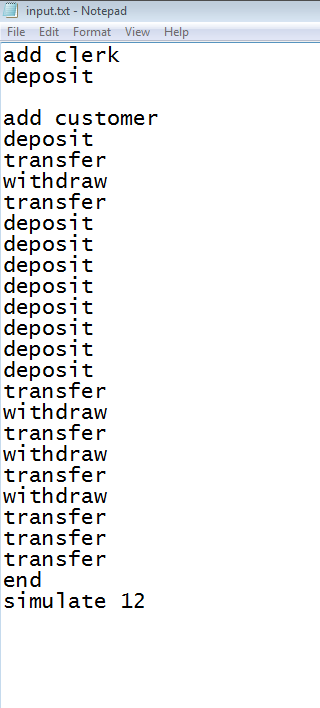
* Press any key to proceed.

## Choose input type



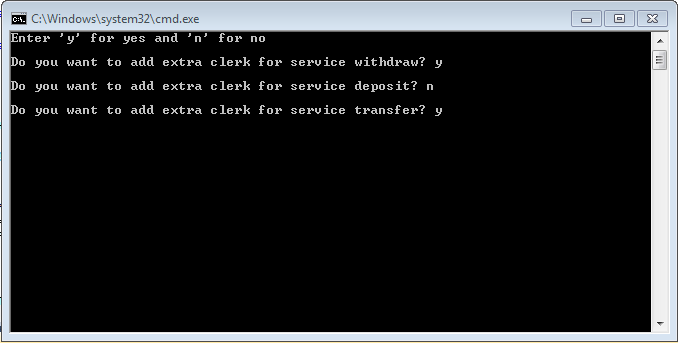
### From file

1. You must write each instruction in a separate line in the file “input.txt”.
2. If you want to add one extra clerk you can type any of these keywords: “Add Clerk" or "add clerk" or "Add clerk" or "add Clerk".
3. Then you must write in new line the service you want to add you can type any of these keywords: “withdraw” or ” Withdraw”, “deposit” or ”Deposit”, ”transfer” or “ Transfer”.
4. Then you will have to enter customers by typing any of these keywords: "add customer" or "Add Customer" or "Add customer".
5. Then enter your customer’s service: “withdraw” or ” Withdraw”, “deposit” or ”Deposit”, ”transfer” or “ Transfer”.
6. When you finish adding customers type “end”.
7. If you want to simulate in a certain instance for all the services you have to write “simulate” then make a “space” and write the value of time you want to simulate at. (Output will be in the file outputs).
8. Here’s a typical example for the input file:



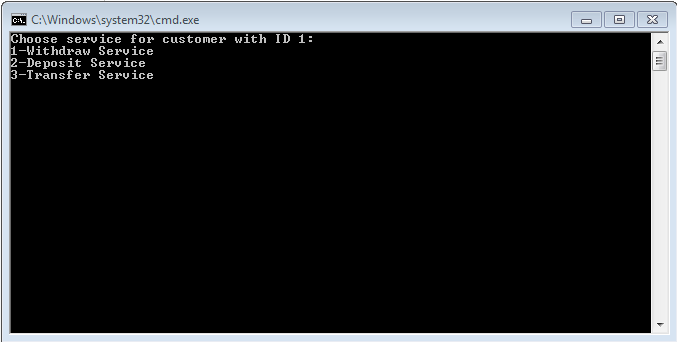
### From console

#### Adding extra clerks

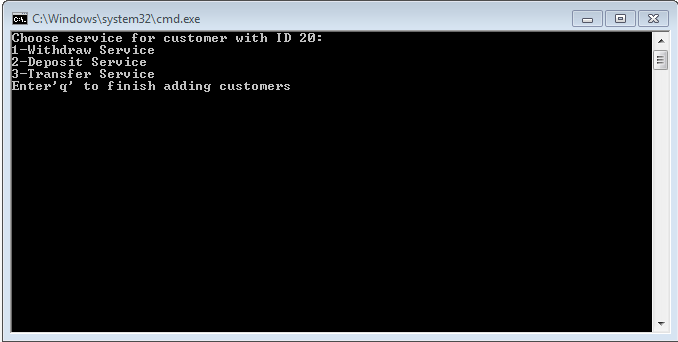


1. Enter ‘y’ or ‘n’ to specify if you want to add the extra clerk for this service or not.

#### Add customers



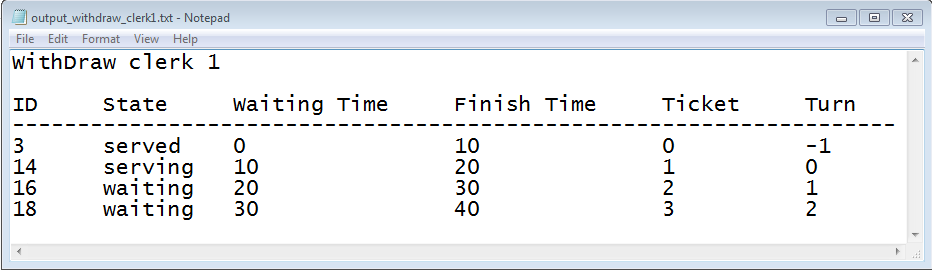
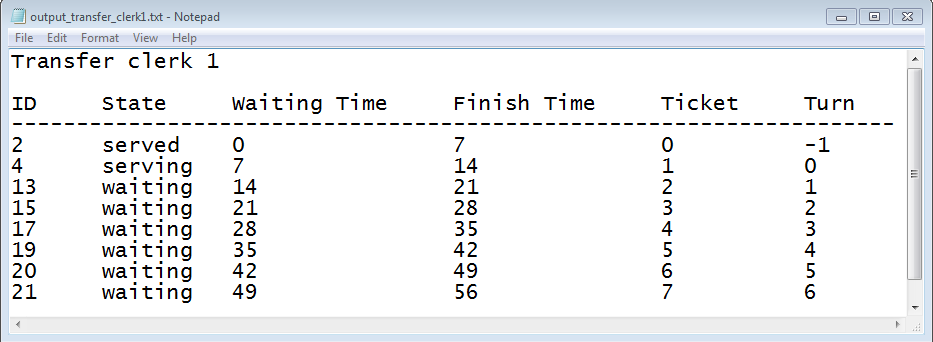
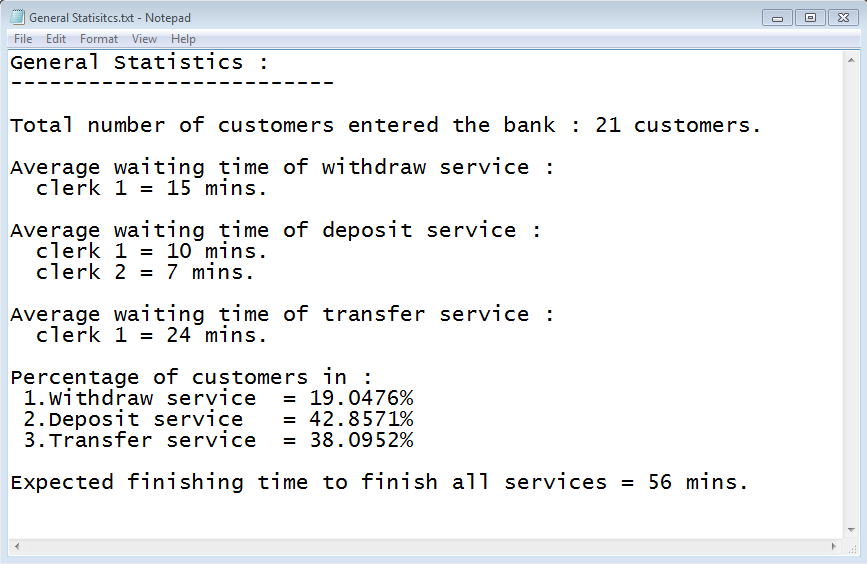
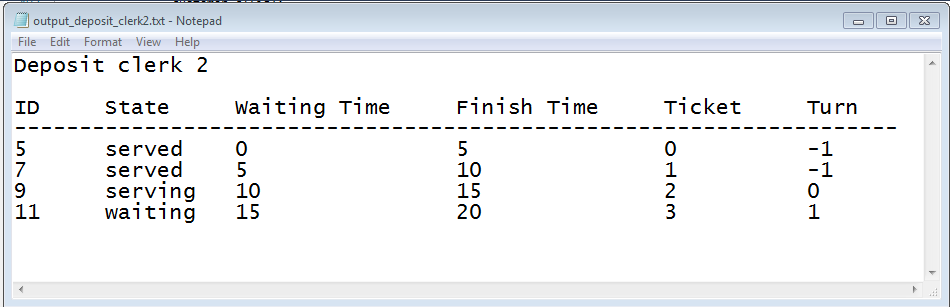
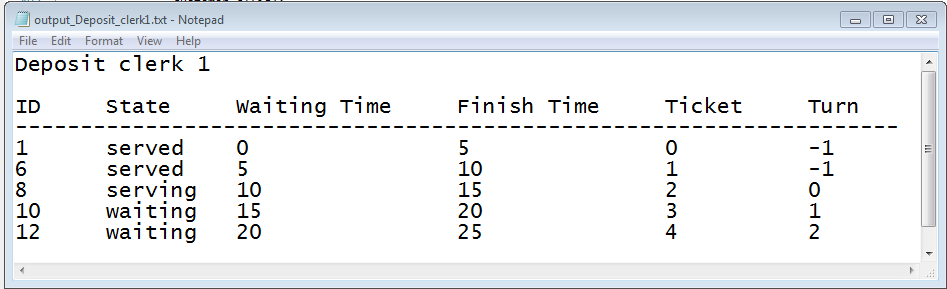
* You must at least enter one customer before going to simulation so you have to choose one of these services.
* Entering customers will continue until you enters ‘q’



## output files

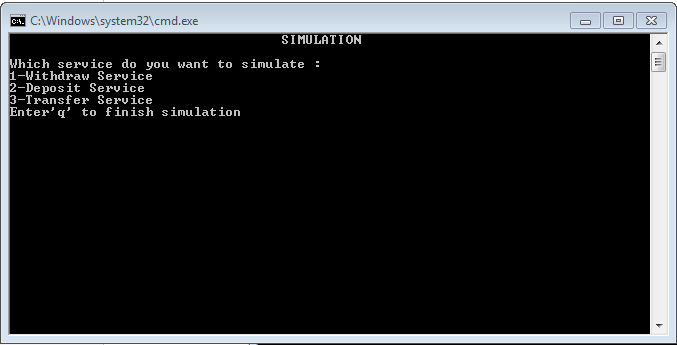
* There will be output file for each service clerk1 and there will be a general statistics file also.
* According to the extra clerks specified from previous steps there will be extra output files generated.

### Typical Output of the file input



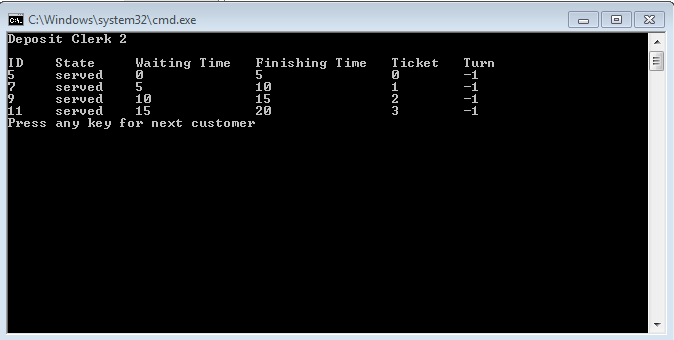
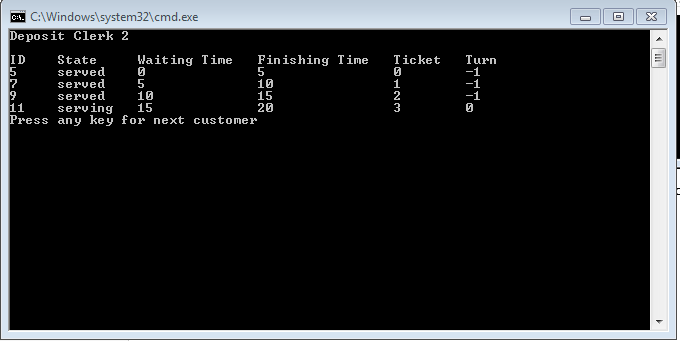
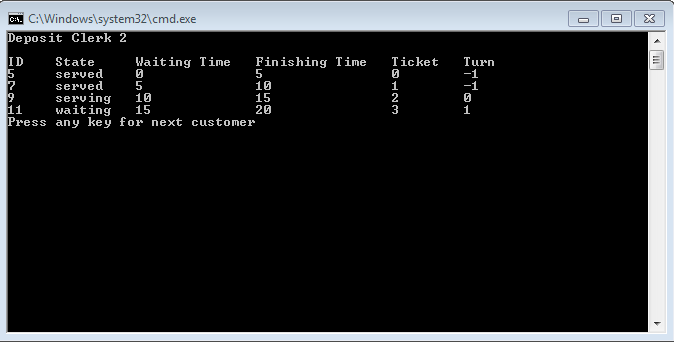
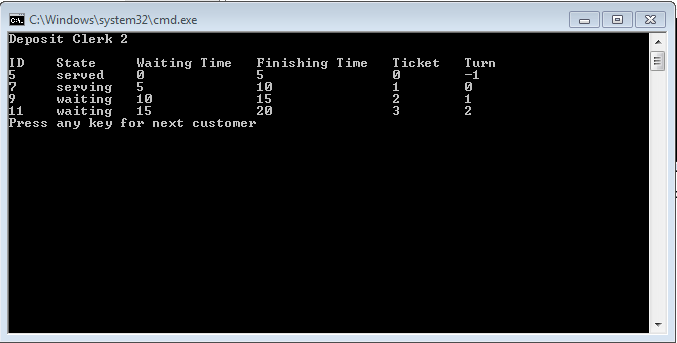
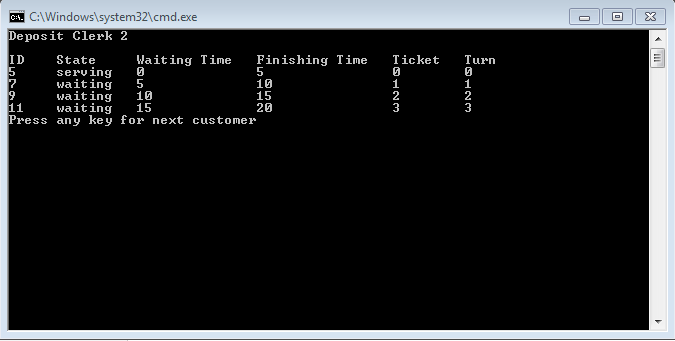
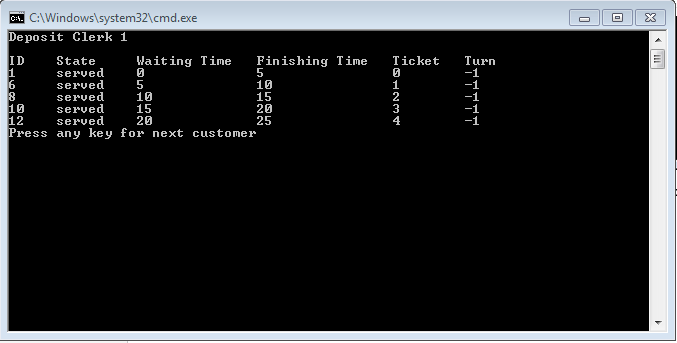
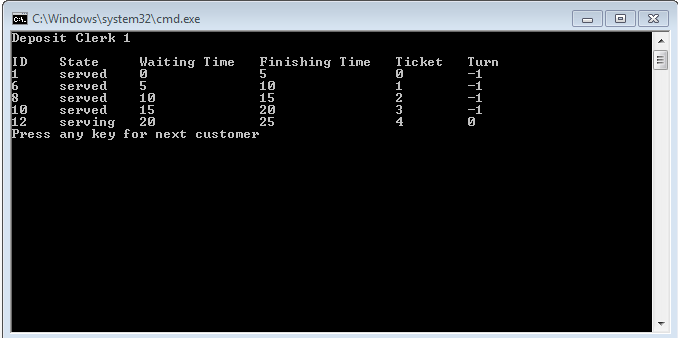
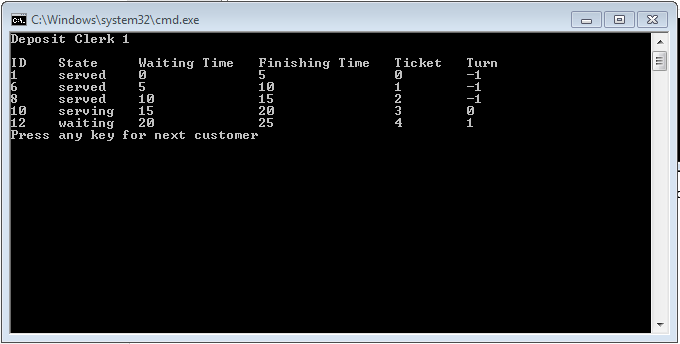
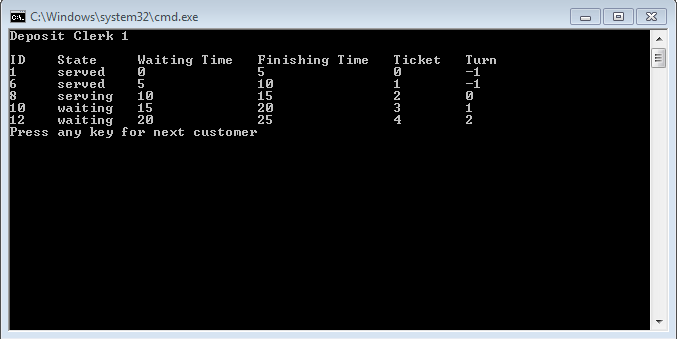
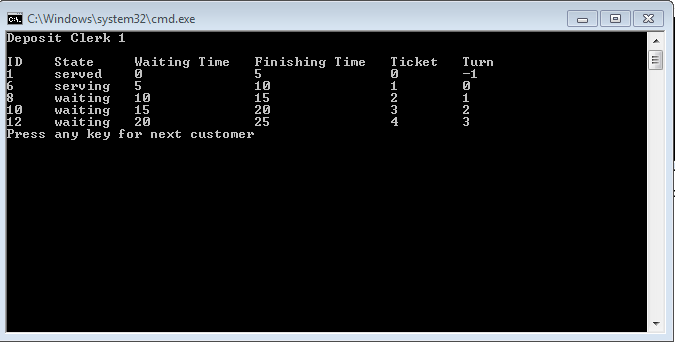
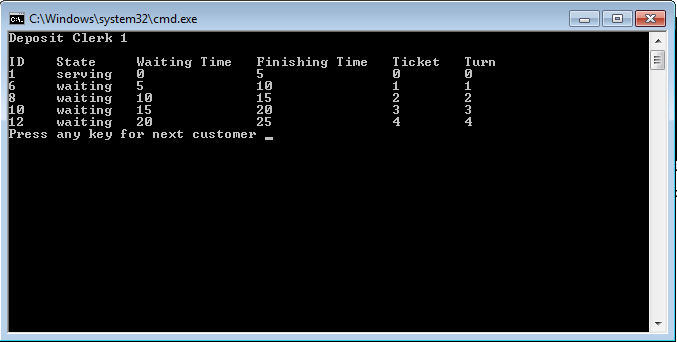
* Noted that this simulation is at instance 12 for all services.

## Simulation



1. Choose one of the service to simulate, noted that if it has 2 clerks then you will see 2 simulations for each clerk concurrently.
2. Press any key to finish the current customer’s service and let the next customer to enter.
3. Enter ‘q’ to finish the simulation and the program.

### Typical simulation for input from file deposit service



# The code

## node.h

template<class T> class node

{

T data;

node<T> \* next;

public:

node() :data(0), next(nullptr) {}

node(T d) :data(d), next(nullptr) {}

T getData() { return data; }

void setData(T d){ data = d; }

node<T> \* getNext() { return next; }

void setNext(node<T> \* n) { next = n; }

template<class T> friend class linkedlist;

};

## Linked list.h

template<class T> class linkedlist

{

private:

node<T> \* chain;

node<T> \* tail;

int size;

public:

linkedlist();

void addAtend(T item); //add at end

T removeAtfront();

int getSize(){ return size; }

};

template<class T> linkedlist<T>::linkedlist()

{

chain = nullptr;

tail = nullptr;

size = 0;

}

template<class T> void linkedlist<T>::addAtend(T item) // add at end

{

node<T> \* temp=new node<T>(item);

temp->next=nullptr;

if (chain == nullptr)

{

chain = temp;

tail = temp;

size++;

}

else

{

tail->next=temp;

tail=temp;

size++;

}

}

template<class T> T linkedlist<T> ::removeAtfront()

{

if (chain == nullptr) return T();

node<T> \* p = chain;

chain = chain->next;

size--;

return p->data ;

}

## customer.h

#include <string.h>

class customer

{

protected:

int ID;

string state;

int waitTime;

int finishingTime;

int ticket;

int turn;

public:

customer() { state = "waiting"; waitTime = 0; finishingTime = 0; ticket = 0; turn = 0; }

customer(int x){ID=x;}

customer(int x, string s, int wt, int ft, int tkt, int t){ ID = x; state = s; waitTime = wt; finishingTime = ft; ticket = tkt; turn = t; }

customer &customer:: operator=(customer &c)

{

ID=c.ID;

state=c.state;

waitTime=c.waitTime;

finishingTime=c.finishingTime;

ticket=c.ticket;

turn=c.turn;

return \*this;

}

void setID(int id) { ID = id; }

int getID() { return ID; }

void setState(string s) { state = s; }

string getState() { return state; }

void setWaitTime(int wt) { waitTime = wt; }

int getWaitTime() { return waitTime; }

void setFinishingTime(int ft) { finishingTime = ft; }

int getFinishingTime() { return finishingTime; }

void setTicket(int tkt) { ticket = tkt; }

int getTicket() { return ticket; }

void setTurn(int t) { turn = t; }

int getTurn() { return turn; }

};

## queue.h

#include "nodeANDlist.h"

#include "customer.h"

template<class T> class queue

{

private:

linkedlist<T> q;

public:

queue() { }

void enqueue(T object) { q.addAtend(object); }

T dequeue() {return q.removeAtfront(); }

int size() { return q.getSize(); }

//bool isEmpty() { return q.isEmpty(); };

};

## clerk.h

#include"queue.h"

#include <string.h>

class clerk

{

private: string service;

int time;

int ID;

int totalWaiting;

queue<customer> Q;

public:

clerk(){service=" ",time=0; ID=1;totalWaiting=0;}

clerk(string s, int t,int id)

{

service=s;

time=t;

ID=id;

}

void setService(string s)

{service=s;}

string getService()

{return service;}

void setTime(int t)

{time=t;}

int getTime()

{return time;}

void setID(int t)

{ID=t;}

int getID()

{return ID;}

void setTotalWaiting(int t)

{totalWaiting=t;}

int getTotalWaiting()

{return totalWaiting;}

int getQueueSize()

{return Q.size();}

void AddCustomer(customer c)

{

Q.enqueue(c);

}

customer removeCustomer()

{

return Q.dequeue();

}

};

## main .cpp

### Initialization and global variables

#include <iostream>

#include <string>

#include<iomanip>

#include<fstream>

#include"clerk.h"

#include<conio.h>

using namespace std;

void ExtraClerk();

void enterCustomer();

void selectSimulate();

void chooseInput();

void statistics(string filename);

void readFile(string fileName);

void setCustomerData(clerk &c);

void SetData\_to\_file(clerk&c);

void draw\_Tobank();

void draw\_Simulation();

void draw\_welcome();

void writeToFile(string filename,clerk & cl);

template<typename T> void printElement(T t,int width);

void printData(clerk &c);

void max\_finish(int x);

bool extra\_Clerk\_Withdraw=false;

bool extra\_Clerk\_Deposit=false;

bool extra\_Clerk\_Transfer=false;

bool SimulateFlag=false;

int number\_of\_customers=0;

int finish\_all\_service=0;

int Average\_waiting\_time=0;

int Simulate\_time=0;

clerk c\_withdraw1("WithDraw",10,1)

,c\_withdraw2("WithDraw",10,2)

,c\_deposit1("Deposit",5,1)

,c\_deposit2("Deposit",5,2)

,c\_transfer1("Transfer",7,1)

,c\_transfer2("Transfer",7,2);

### int main()

{

remove("output\_withdraw\_clerk1.txt");

remove("output\_withdraw\_clerk2.txt");

remove("output\_Deposit\_clerk1.txt");

remove("output\_Deposit\_clerk2.txt");

remove("output\_transfer\_clerk1.txt");

remove("output\_transfer\_clerk2.txt");

draw\_Tobank();

draw\_Simulation();

cout<<"press any key to enter "<<endl;

getch();

system("CLS");

chooseInput();

SetData\_to\_file(c\_withdraw1);

writeToFile("output\_withdraw\_clerk1.txt",c\_withdraw1);

SetData\_to\_file(c\_deposit1);

writeToFile("output\_Deposit\_clerk1.txt",c\_deposit1);

SetData\_to\_file(c\_transfer1);

writeToFile("output\_transfer\_clerk1.txt",c\_transfer1);

if(extra\_Clerk\_Withdraw)

{

SetData\_to\_file(c\_withdraw2);

writeToFile("output\_withdraw\_clerk2.txt",c\_withdraw2);

}

if(extra\_Clerk\_Deposit)

{

SetData\_to\_file(c\_deposit2);

writeToFile("output\_deposit\_clerk2.txt",c\_deposit2);

}

if(extra\_Clerk\_Transfer)

{

SetData\_to\_file(c\_transfer2);

writeToFile("output\_transfer\_clerk2.txt",c\_transfer2);

}

statistics("General Statisitcs.txt");

selectSimulate();

return 0;

}

### void ExtraClerk()

{

string x1,x2,x3;

cout<<"Enter 'y' for yes and 'n' for no\n";

do{

cout<<"\nDo you want to add extra clerk for service withdraw? ";

cin>>x1;

}

while(x1!="y"&& x1!="n");

do{

cout<<"\nDo you want to add extra clerk for service deposit? ";

cin>>x2;

}

while(x2!="y"&& x2!="n");

do{

cout<<"\nDo you want to add extra clerk for service transfer? ";

cin>>x3;

}

while(x3!="y"&& x3!="n");

cout<<endl;

if(x1=="y")

extra\_Clerk\_Withdraw=true;

if(x2=="y")

extra\_Clerk\_Deposit=true;

if(x3=="y")

extra\_Clerk\_Transfer=true;

system("CLS"); //clear the screen

}

### void enterCustomer ()

{

int id=1;

string input;

int toggle\_withdraw=1;

int toggle\_Deposit=1;

int toggle\_transfer=1;

do{

do{

cout<<"Choose service for customer with ID "<<id<<":"<<endl;

cout<<"1-Withdraw Service"<<endl;

cout<<"2-Deposit Service"<<endl;

cout<<"3-Transfer Service"<<endl;

if(id!=1)

cout<<"Enter'q' to finish adding customers"<<endl;

cin>>input;

system("CLS");

}

while(input!="1" && input!="2" && input!="3" && input!="q");

if(input=="1")

{

if(extra\_Clerk\_Withdraw==true)

{

if(toggle\_withdraw)

{

customer c1(id);

c\_withdraw1.AddCustomer(c1);

toggle\_withdraw=1-toggle\_withdraw;

}

else

{

customer c1(id);

c\_withdraw2.AddCustomer(c1);

toggle\_withdraw=1-toggle\_withdraw;

}

}

else

{

customer c1(id);

c\_withdraw1.AddCustomer(c1);

}

}

else if(input=="2")

{

if(extra\_Clerk\_Deposit==true)

{

if(toggle\_Deposit)

{

customer c1(id);

c\_deposit1.AddCustomer(c1);

toggle\_Deposit=1-toggle\_Deposit;

}

else

{

customer c1(id);

c\_deposit2.AddCustomer(c1);

toggle\_Deposit=1-toggle\_Deposit;

}

}

else

{

customer c1(id);

c\_deposit1.AddCustomer(c1);

}

}

else if(input=="3")

{

if(extra\_Clerk\_Transfer==true)

{

if(toggle\_transfer)

{

customer c1(id);

c\_transfer1.AddCustomer(c1);

toggle\_transfer=1-toggle\_transfer;

}

else

{

customer c1(id);

c\_transfer2.AddCustomer(c1);

toggle\_transfer=1-toggle\_transfer;

}

}

else

{

customer c1(id);

c\_transfer1.AddCustomer(c1);

}

}

id++;

system("CLS");

}while(input!="q");

}

### void selectSimulate()

{

string select;

do

{

do

{

cout<<" SIMULATION "<<endl<<endl;

cout<<"Which service do you want to simulate : "<<endl;

cout<<"1-Withdraw Service"<<endl;

cout<<"2-Deposit Service"<<endl;

cout<<"3-Transfer Service"<<endl;

cout<<"Enter'q' to finish simulation"<<endl;

cin>>select;

system("CLS");

}

while(select!="1" && select!="2" && select!="3" && select!="q");

if(select=="1")

{

if(extra\_Clerk\_Withdraw)

{

printData(c\_withdraw1);

printData(c\_withdraw2);

}

else

printData(c\_withdraw1);

}

else if(select=="2")

{

if(extra\_Clerk\_Deposit)

{

printData(c\_deposit1);

printData(c\_deposit2);

}

else

printData(c\_deposit1);

}

else if (select=="3")

{

if(extra\_Clerk\_Transfer)

{

printData(c\_transfer1);

printData(c\_transfer2);

}

else

printData(c\_transfer1);

}

}

while(select!="q");

}

### void chooseInput()

{string x1;

do{

draw\_welcome();

cout<<"Choose input type \n";

cout<<"1)from file\n";

cout<<"2)Enter the data in the console\n";

cin>>x1;

if(x1=="1")

{

system("CLS");

readFile("input.txt");

}

else if(x1=="2")

{

system("CLS");

ExtraClerk();

enterCustomer ();

}

system("CLS");

}while(x1!="1" && x1!="2");

}

### void printData(clerk &c)

{

int x=c.getQueueSize();

do{

cout<<c.getService()<<" "<<"Clerk "<<c.getID()<<endl<<endl;

printElement("ID",6);

printElement("State",10);

printElement("Waiting Time",15);

printElement("Finishing Time",17);

printElement("Ticket",9);

printElement("Turn",7);

cout<<endl;

for(int i=0;i<c.getQueueSize();i++)

{

customer c1=c.removeCustomer();

printElement(c1.getID(),6);

printElement(c1.getState(),10);

printElement(c1.getWaitTime(),15);

printElement(c1.getFinishingTime(),17);

printElement(c1.getTicket(),9);

printElement(c1.getTurn(),7);

cout<<endl;

if(c1.getTurn()!=-1)

c1.setTurn(c1.getTurn()-1);

switch(c1.getTurn())

{

case -1:c1.setState("served");break;

case 0:c1.setState("serving");break;

default:c1.setState("waiting");break;

}

c.AddCustomer(c1);

}

cout<<"Press any key for next customer ";

getch();

system("CLS");

x--;

if(x<0)

break;

}while(true);

setCustomerData(c);

}

### void setCustomerData(clerk &c)

{

int f=0;

int sum=0;

for(int i=0;i<c.getQueueSize();i++)

{

customer c1=c.removeCustomer();

if(i==0)

c1.setState("serving");

else

c1.setState("waiting");

c1.setWaitTime(i\*c.getTime());

c1.setFinishingTime(c1.getWaitTime()+c.getTime());

c1.setTicket(i);

c1.setTurn(i);

sum+=c1.getWaitTime();

c.setTotalWaiting(sum);

f=c1.getFinishingTime();

max\_finish(f);

c.AddCustomer(c1);

}

}

### template<typename T> void printElement(T t,int width)

{

cout<<left<<setw(width)<<setfill(' ')<<t;

}

### void readFile(string fileName)

{

int toggle\_withdraw=1;

int toggle\_Deposit=1;

int toggle\_transfer=1;

int id=1;

ifstream infile;

infile.open(fileName);

bool addClerk=false;

bool clerkW=false;

bool clerkD=false;

bool clerkT=false;

string line;

getline(infile,line);

if(line=="Add Clerk"||line=="add clerk"||line=="Add clerk"||line=="add Clerk")

{

addClerk=true;

string clerk;

while(true)

{

getline(infile,clerk);

if(clerk=="")

continue;

if(clerk=="withdraw"||clerk=="Withdraw")

clerkW=true;

else if(clerk=="deposit"||clerk=="Deposit")

clerkD=true;

else if(clerk=="transfer"||clerk=="Transfer")

clerkT=true;

else if(clerk=="add customer"||clerk=="Add Customer"||clerk=="Add customer")

break;

else

{

cout<<"Error.Cannot read from file";

break;

}

}

}

string cu;

while(true)

{

getline(infile,cu);

if(cu=="")

continue;

if(cu=="withdraw"||cu=="Withdraw")

{

if(addClerk&&clerkW)

{

extra\_Clerk\_Withdraw=true;

if(toggle\_withdraw)

{

customer c1(id);

c\_withdraw1.AddCustomer(c1);

toggle\_withdraw=1-toggle\_withdraw;

}

else

{

customer c1(id);

c\_withdraw2.AddCustomer(c1);

toggle\_withdraw=1-toggle\_withdraw;

}

}

else

{

customer c1(id);

c\_withdraw1.AddCustomer(c1);

}

}

else if(cu=="deposit"||cu=="Deposit")

{

if(addClerk&&clerkD)

{

extra\_Clerk\_Deposit=true;

if(toggle\_Deposit)

{

customer c1(id);

c\_deposit1.AddCustomer(c1);

toggle\_Deposit=1-toggle\_Deposit;

}

else

{

customer c1(id);

c\_deposit2.AddCustomer(c1);

toggle\_Deposit=1-toggle\_Deposit;

}

}

else

{

customer c1(id);

c\_deposit1.AddCustomer(c1);

}

}

else if(cu=="transfer"||cu=="Transfer")

{

if(addClerk&&clerkT)

{

extra\_Clerk\_Transfer=true;

if(toggle\_transfer)

{

customer c1(id);

c\_transfer1.AddCustomer(c1);

toggle\_transfer=1-toggle\_transfer;

}

else

{

customer c1(id);

c\_transfer2.AddCustomer(c1);

toggle\_transfer=1-toggle\_transfer;

}

}

else

{

customer c1(id);

c\_transfer1.AddCustomer(c1);

}

}

else if(cu=="end")

break;

else

{

cout<<"Error.Cannot read from file";

break;

}

id++;

}

while(!infile.eof())

{

string sim;

infile>>sim;

if(sim=="simulate")

infile>>Simulate\_time;

}

}

### void writeToFile(string filename,clerk &c)

{

ofstream outfile;

outfile.open(filename);

outfile<<c.getService()<<" "<<"clerk "<<c.getID()<<endl<<endl;

outfile<<left<<setw(7)<<setfill(' ')<<"ID";

outfile<<left<<setw(10)<<setfill(' ')<<"State";

outfile<<left<<setw(17)<<setfill(' ')<<"Waiting Time";

outfile<<left<<setw(16)<<setfill(' ')<<"Finish Time";

outfile<<left<<setw(11)<<setfill(' ')<<"Ticket";

outfile<<left<<setw(7)<<setfill(' ')<<"Turn";

outfile<<endl;

outfile<<"--------------------------------------------------------------------"<<endl;

for(int i=0;i<c.getQueueSize();i++)

{

customer c1=c.removeCustomer();

outfile<<left<<setw(7)<<setfill(' ')<<c1.getID();

outfile<<left<<setw(10)<<setfill(' ')<<c1.getState();

outfile<<left<<setw(17)<<setfill(' ')<<c1.getWaitTime();

outfile<<left<<setw(16)<<setfill(' ')<<c1.getFinishingTime();

outfile<<left<<setw(11)<<setfill(' ')<<c1.getTicket();

outfile<<left<<setw(7)<<setfill(' ')<<c1.getTurn();

outfile<<endl;

c.AddCustomer(c1);

}

setCustomerData(c);

}

### void SetData\_to\_file(clerk&c)

{

int k=Simulate\_time/(c.getTime());

for(int i=0;i<c.getQueueSize();i++)

{

customer c1=c.removeCustomer();

if(i<k)

{

c1.setState("served");

c1.setTurn(-1);

}

else if(i==k)

{

c1.setState("serving");

c1.setTurn(0);

}

else

{

c1.setState("waiting");

c1.setTurn(i-k);

}

c1.setWaitTime(i\*c.getTime());

c1.setFinishingTime(c1.getWaitTime()+c.getTime());

c1.setTicket(i);

c.AddCustomer(c1);

}

}

### void draw\_Tobank()

{

cout<<" 0000000 0000 000 00 00 00 "<<endl;

cout<<" 00 00 00 00 0000 00 00 00 "<<endl;

cout<<" 000000 00 00 00 00 00 00 0000 "<<endl;

cout<<" 00 00 00 00 00 00 00 00 00 "<<endl;

cout<<" 0000000 00 00 00 0000 00 00 "<<endl<<endl<<endl;

}

### void draw\_Simulation()

{

cout<<" 00000 0 0 0 0 0 0 0000 000000 0 00000 00 0"<<endl;

cout<<" 00 00 00 00 0 0 0 00 00 00 00 00 00 00 0"<<endl;

cout<<" 00000 00 0 00 0 0 0 0 00 00 00 00 00 00 00 0 0 0"<<endl;

cout<<" 00 00 0 0 0 0 0 00 00 00 00 00 00 0 0 0"<<endl;

cout<<"00000 00 0 0 00000 000000 00 00 00 00 00000 0 00"<<endl<<endl<<endl;

}

### void draw\_welcome()

{

cout<<" 00 00 000000 00 0000000 00000 00 00 000000 "<<endl;

cout<<" 00 00 00 00 00 00 00 00 000 000 00 "<<endl;

cout<<" 00 00 00 0000 00 00 00 00 00 00 00 0000 "<<endl;

cout<<" 00 00 00 00 00 00 00 00 00 00 00 "<<endl;

cout<<" 000 000 000000 0000000 0000000 00000 00 00 000000 "<<endl<<endl<<endl;

}

### void max\_finish(int x)

{

if(x>=finish\_all\_service)

finish\_all\_service=x;

}

### void statistics(string filename)

{

ofstream outfile;

outfile.open(filename);

number\_of\_customers=c\_withdraw1.getQueueSize()+c\_withdraw2.getQueueSize()+c\_deposit1.getQueueSize()

+c\_deposit2.getQueueSize()+c\_transfer1.getQueueSize()+c\_transfer2.getQueueSize();

outfile<<"General Statistics : "<<endl;

outfile<<"-------------------------"<<endl<<endl;

outfile<<"Total number of customers entered the bank : " <<number\_of\_customers<<" customers."<<endl<<endl;

outfile<<"Average waiting time of withdraw service :"<<endl;

if(c\_withdraw1.getQueueSize()!=0)

outfile<<" clerk 1 = "<<c\_withdraw1.getTotalWaiting()/c\_withdraw1.getQueueSize()<<" mins."<<endl;

else

outfile<<" clerk 1 = 0 mins."<<endl;

if(extra\_Clerk\_Withdraw && c\_withdraw2.getQueueSize()!=0)

outfile<<" clerk 2 = " <<c\_withdraw2.getTotalWaiting()/c\_withdraw2.getQueueSize()<<" mins."<<endl;

else if(c\_withdraw2.getQueueSize()==0)

outfile<<" clerk 2 = 0 mins."<<endl;

outfile<<endl;

outfile<<"Average waiting time of deposit service :"<<endl;

if(c\_deposit1.getQueueSize()!=0)

outfile<<" clerk 1 = "<<c\_deposit1.getTotalWaiting()/c\_deposit1.getQueueSize()<<" mins."<<endl;

else

outfile<<" clerk 1 = 0 mins."<<endl;

if(extra\_Clerk\_Deposit &&c\_deposit2.getQueueSize()!=0)

outfile<<" clerk 2 = " <<c\_deposit2.getTotalWaiting()/c\_deposit2.getQueueSize()<<" mins."<<endl;

else if (c\_deposit2.getQueueSize()==0)

outfile<<" clerk 2 = 0 mins."<<endl;

outfile<<endl;

outfile<<"Average waiting time of transfer service :"<<endl;

if(c\_transfer1.getQueueSize()!=0)

outfile<<" clerk 1 = " <<c\_transfer1.getTotalWaiting()/c\_transfer1.getQueueSize()<<" mins."<<endl;

else

outfile<<" clerk 1 = 0 mins."<<endl;

if(extra\_Clerk\_Transfer&&c\_transfer2.getQueueSize()!=0)

outfile<<" clerk 2 = " <<c\_transfer2.getTotalWaiting()/c\_transfer2.getQueueSize()<<" mins."<<endl;

else if (c\_transfer2.getQueueSize()==0)

outfile<<" clerk 2= 0 mins."<<endl;

outfile<<endl;

outfile<<"Percentage of customers in : "<<endl;

outfile<<" 1.Withdraw service = "<<(((c\_withdraw1.getQueueSize()+c\_withdraw2.getQueueSize())\*1.0)/number\_of\_customers)\*100.0<<"%"<<endl;

outfile<<" 2.Deposit service = "<<(((c\_deposit1.getQueueSize()+c\_deposit2.getQueueSize())\*1.0)/number\_of\_customers)\*100.0<<"%"<<endl;

outfile<<" 3.Transfer service = "<<(((c\_transfer1.getQueueSize()+c\_transfer2.getQueueSize())\*1.0)/number\_of\_customers)\*100.0<<"%"<<endl;

outfile<<endl;

outfile<<"Expected finishing time to finish all services = "<<finish\_all\_service <<" mins."<<endl;

}