

**UNIVERSITI TEKNOLOGI MALAYSIA, JOHOR BAHRU**

**SEMESTER I - 2019/2020**

**SCSJ 2013 – DATA STRUCTURE AND ALGORITHM**



Queue Management System for Campus Election in University of Technology, Malaysia



**GROUP NAME :** Evil Geniuses

**LECTURER’ NAME :** Dr. Ruhaidah binti Samsudin

**GROUP MEMBERS :**

|  |  |  |
| --- | --- | --- |
| 1) | AZMIL AZIZI BIN NORDIN | A18CS0041 |
| 2) | MOHD AINAL FARHAN BIN MOHAMAD JOHARI | A18CS0113 |
| 3) | MOHAMMAD RIDZWAN SYAH BIN IRWAN | A18CS0112 |
| 4) | MOHAMAD FAIZ HAKIMI BIN ISHAK | A18CS0107 |

## Objective of the mini Project.

This project plans to execute campus elections in university with more manageable and time-savvy way. The mini Project that we are proposing are mainly based on the listed objectives below:

* + Ensuring orderly flow of voters in taking turns for voting process.
  + Making sure that queuing does left students unoccupied.
  + Decrease area crowd.

## Synopsis of the mini Project.

* + Project Title :

Queue Management System for Campus Election in University of Technology, Malaysia.

* Domain :

Data Structure and Algorithm – Queue

* Problem statement :

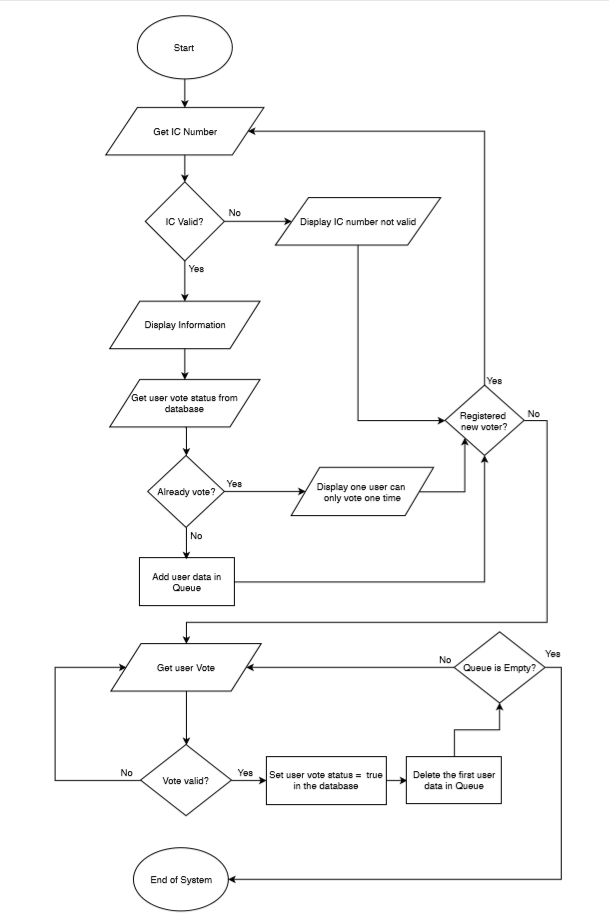
Taking turns for voting are usually cause time inefficiency and crowded environment.

* Project description :

The system to be developed is on Queue Management. In this system, we are implementing queue using linked lists. There will be 2 users of the system, the election officer and the voter. The election officers should be able to facilitate the queueing process while have full knowledge of the system control. In the other hand, the voters should be able to key in details in the Queue Management System.

When the voter first arrives for voting, the election officers will brief the procedures of queuing using the proposed system. Voters will then be instructed to key in details. The voter’s details will be used for turn taking, and data recording. If the queue is full, the voters will be able to key in their details in advance and when their turn is up, they will be notified to show up.

* Flowchart



* Case Diagram

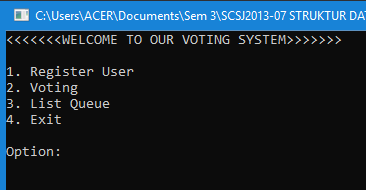
## 

## UML Class Diagram

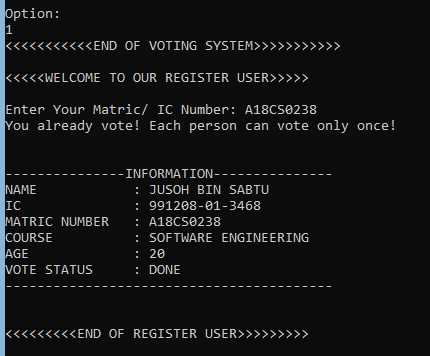
|  |
| --- |
| QueueNode |
| +QueueNode(): QueueNode  +~QueueNode(): QueueNode  +name: string  +ic: string  +matricNo: string  +course: string  +int age: int  + voteStatus: bool  +next: QueueNode\* |

|  |
| --- |
| Queue |
| +count: int |
| -name: string\*  -ic: string\*  -matricNo: string\*  -course: string\*  -age: int\*  -voteStatus: bool\* |
| +countCan: int  +canName: string\*  +canYear: string\*  +canFaculty: string\*  +canTargetedPosition: string\*  +canManifesto: string\*  +canVote: int\* |
| +Queue (): Queue  +~Queue (): Queue  +front: QueueNode\*  +rear: QueueNode\*  +getName (index: int)const: string  +getIC (index: int)const: string  +getMatricNo (index: int)const: string  +getCourse (index: int)const: string  +getAge (index: int)const: int  +getVoteStatus (index: int)const: bool  +setVoteStatus(status: bool): void  +updateFile(): void |

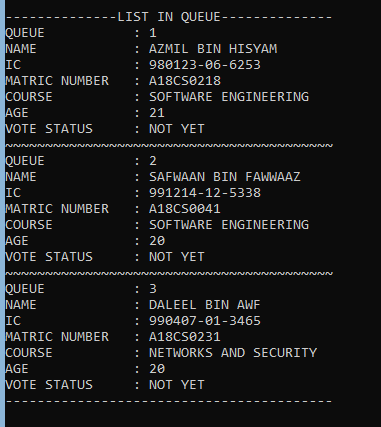
## System Prototype



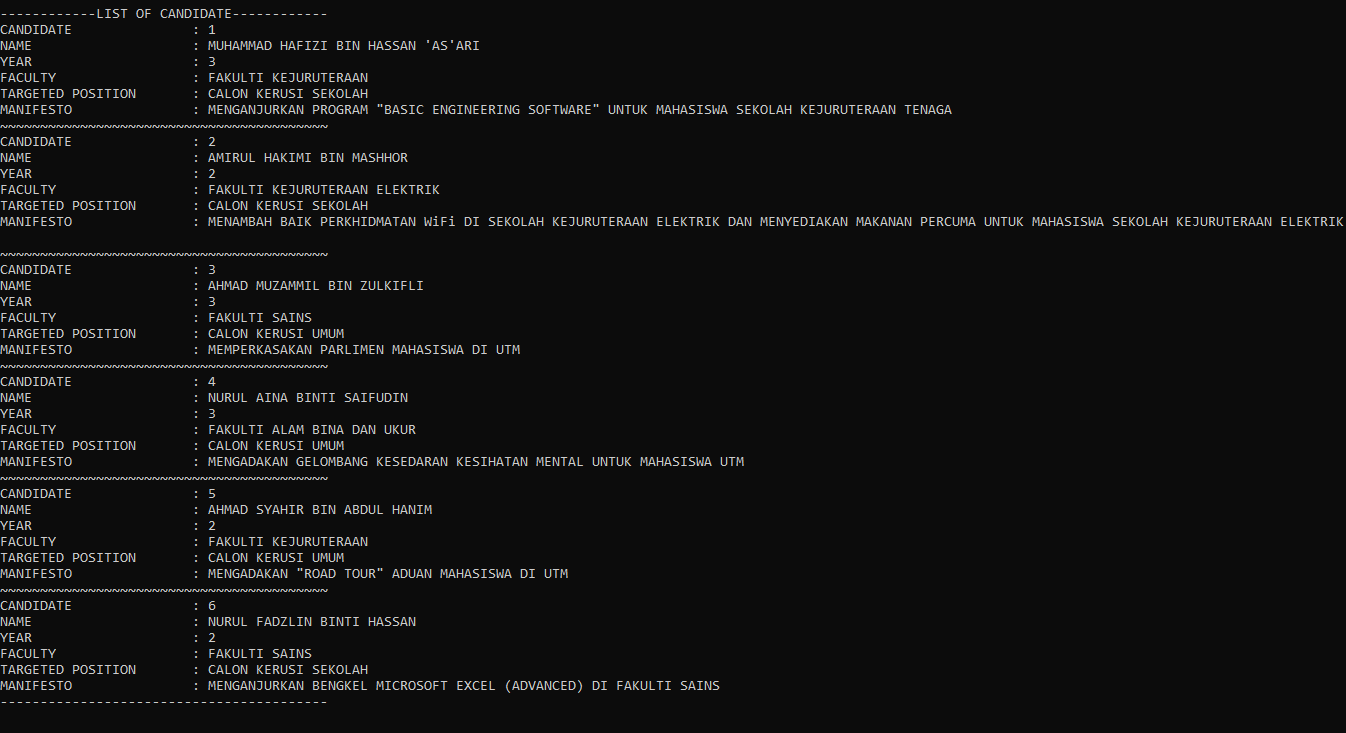
Main menu of the Voting System



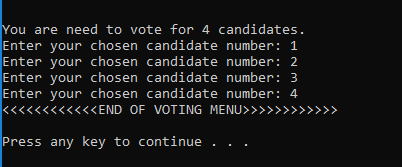
Registering the user



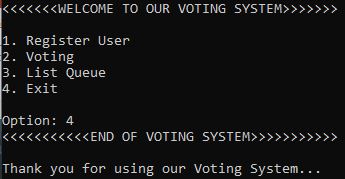
Check the queue



List of candidate



Voting process

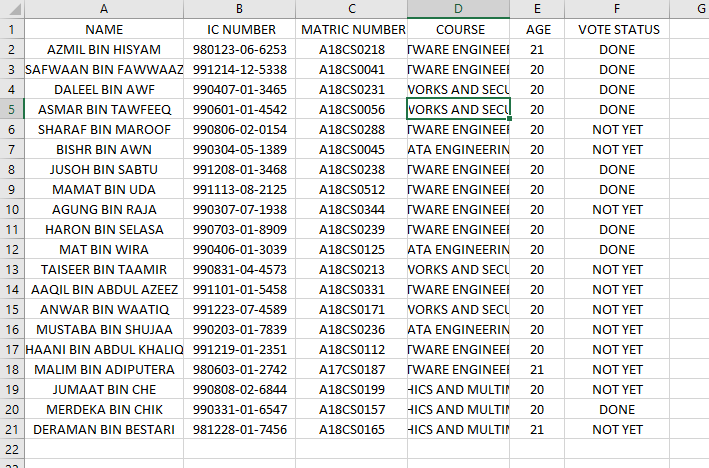


Exit

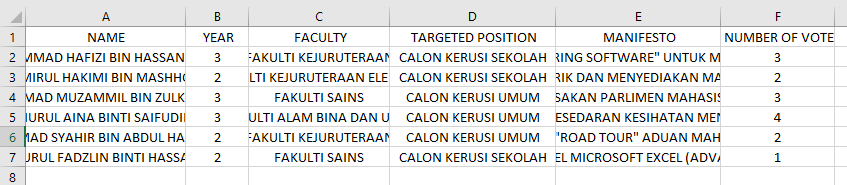
## Development Activities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Meeting Date | Members Participate in the meeting | Activity | Task for each member | Task Achieved (yes/no) |
| 10/12/2019 | Ainal  Ridzwan  Azmil  Faiz | Discussion on Assignment part 1-Sorting | UML case diagram - Azmil  Source Code – Ainal, Ridzwan, Azmil, Faiz  Report - Faiz | yes |
| 13/12/2019 | Ainal  Ridzwan  Azmil  Faiz | Discussion on Assignment part 2 - Implementation on Linked list | UML case diagram-Faiz  Source Code - Ainal, Ridzwan, Azmil, Faiz  Report - Ridzwan | yes |
| 14/12/2019 | Ainal  Ridzwan  Azmil  Faiz | Discussion on Project DSA- implementation on Queue or Stack | UML case diagram-Faiz  Source Code - Ainal, Ridzwan, Azmil, Faiz  Report - Ainal | yes |

## Appendix



Database File



Candidate File

## Coding

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

using namespace std;

class QueueNode {

public:

QueueNode();

~QueueNode();

string name;

string ic;

string matricNo;

string course;

int age;

bool voteStatus;

QueueNode\* next;

};

QueueNode::QueueNode(){}

QueueNode::~QueueNode(){}

class Queue{

public:

int count;

private:

string \*name;

string \*ic;

string \*matricNo;

string \*course;

int \*age;

bool \*voteStatus;

public:

int countCan; // Number of Candidate

string\* canName;

int\* canYear;

string\* canFaculty;

string\* canTargetedPosition;

string\* canManifesto;

int\* canVote;

public:

Queue();

~Queue();

QueueNode \*front, \*rear;

string getName (int)const;

string getIC (int)const;

string getMatricNo (int)const;

string getCourse (int)const;

int getAge (int)const;

bool getVoteStatus (int)const;

void setVoteStatus(bool);

void updateFile();

};

Queue::Queue() {

string file = "database.csv", file2 = "candidate.csv";

char c;

count = -1;

countCan = -1;

fstream inF, inFile;

inF.open(file2.c\_str(), ios::in);

inFile.open(file.c\_str(), ios::in);

if(!inFile.is\_open()){

cout << "File failed to open! File Name: " << file << endl;

return;

}

if(!inF.is\_open()){

cout << "File failed to open! File Name: " << file2 << endl;

return;

}

while(inFile.get(c)){

if(inFile.eof())

break;

else if(c == '\n')

count++;

}count++;

while(inF.get(c)){

if(inF.eof())

break;

else if(c == '\n')

countCan++;

}countCan++;

name = new string[count];

ic = new string[count];

matricNo = new string[count];

course = new string[count];

age = new int[count];

voteStatus = new bool[count];

string statusVote;

canName = new string[countCan];

canYear = new int[countCan];

canFaculty = new string[countCan];

canTargetedPosition = new string[countCan];

canManifesto = new string[countCan];

canVote = new int[countCan];

inFile.clear();

inFile.seekg(0L, ios::beg);

while(inFile.get(c)){

if(c == '\n')

break;

}

inF.clear();

inF.seekg(0L, ios::beg);

while(inF.get(c)){

if(c == '\n')

break;

}

for(int i = 0; i < count; i++){

getline(inFile, name[i], ',');

getline(inFile, ic[i], ',');

getline(inFile, matricNo[i], ',');

getline(inFile, course[i], ',');

inFile >> age[i];

inFile.ignore(1, ',');

getline(inFile, statusVote);

if(statusVote == "DONE")

voteStatus[i] = true;

else

voteStatus[i] = false;

}

for(int i = 0; i < count; i++){

getline(inF, canName[i], ',');

inF >> canYear[i];

inF.ignore(1, ',');

getline(inF, canFaculty[i], ',');

getline(inF, canTargetedPosition[i], ',');

getline(inF, canManifesto[i], ',');

inF >> canVote[i];

inF.ignore(1, '\n');

}

inFile.close();

inF.close();

}

Queue::~Queue() {

delete [] name;

delete [] ic;

delete [] matricNo;

delete [] course;

delete [] age;

delete [] voteStatus;

}

string Queue::getName (int index)const{return name[index];}

string Queue::getIC (int index)const{return ic[index];}

string Queue::getMatricNo (int index)const{return matricNo[index];}

string Queue::getCourse (int index)const{return course[index];}

int Queue::getAge (int index)const{return age[index];}

bool Queue::getVoteStatus (int index)const{return voteStatus[index];}

void Queue::setVoteStatus(bool status){

int i = 0;

for(i = 0; i < count; i++){

if(matricNo[i] == front->matricNo)

break;

}

voteStatus[i] = status;

front->voteStatus = status;

}

void Queue::updateFile(){

string file = "database.csv", file2 = "candidate.csv";

fstream outFile, outF;

outFile.open(file.c\_str(), ios::out);

outF.open(file2.c\_str(), ios::out);

outFile << "NAME, IC NUMBER, MATRIC NUMBER, COURSE, AGE, VOTE STATUS" << endl;

for(int i = 0; i < count; i++){

outFile << name[i] << ',';

outFile << ic[i] << ',';

outFile << matricNo[i] << ',';

outFile << course[i] << ',';

outFile << age[i] << ',';

if(voteStatus[i])

outFile << "DONE";

else

outFile << "NOT YET";

if(!(i+1 == count))

outFile << endl;

}

outF << "NAME, YEAR, FACULTY, TARGETED POSITION, MANIFESTO, NUMBER OF VOTE\n";

for(int i = 0; i < countCan; i++){

outF << canName[i] << ",";

outF << canYear[i] << ",";

outF << canFaculty[i] << ",";

outF << canTargetedPosition[i] << ",";

outF << canManifesto[i] << ",";

outF << canVote[i];

if(!(i+1 == countCan))

outF << endl;

}

outF.close();

outFile.close();

}

QueueNode\* registerVoter(Queue\* pointer, int index){

QueueNode\* temp = new QueueNode;

temp->name = pointer->getName(index);

temp->ic = pointer->getIC(index);

temp->matricNo = pointer->getMatricNo(index);

temp->course = pointer->getCourse(index);

temp->age = pointer->getAge(index);

temp->voteStatus = pointer->getVoteStatus(index);

temp->next = NULL;

return temp;

}

bool findVoter(Queue\* pointer, string icOrMatric){

QueueNode\* temp = pointer->rear;

// Declare new variable fc: first character to hold the first character of icOrMatric

char fc = icOrMatric.at(0);

while(temp){

if((fc == 'A') || (fc == 'B') || (fc == 'X')){

if(temp->matricNo == icOrMatric)

return true;

}

else{

if(temp->ic == icOrMatric)

return true;

}

temp = temp->next;

}

return false;

}

void displayQueueList(Queue\* pointer){

int num = 0;

QueueNode\* temp = pointer->front;

if(pointer->front == NULL){

cout << "The Queue is Empty!" << endl;

return;

}

cout << "\n--------------LIST IN QUEUE--------------" << endl;

while(temp){

cout << "QUEUE : " << num+1 << endl;

cout << "NAME : " << temp->name << endl;

cout << "IC : " << temp->ic << endl;

cout << "MATRIC NUMBER : " << temp->matricNo << endl;

cout << "COURSE : " << temp->course << endl;

cout << "AGE : " << temp->age << endl;

cout << "VOTE STATUS : ";

if(temp->voteStatus)

cout << "DONE" << endl;

else

cout << "NOT YET" << endl;

if(temp->next)

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

temp = temp->next;

num++;

}

cout << "-----------------------------------------\n\n" << endl;

}

void displayCandidateList(Queue\* pointer){

cout << "\n------------LIST OF CANDIDATE------------" << endl;

for(int i = 0; i < pointer->countCan; i++){

cout << "CANDIDATE : " << i+1 << endl;

cout << "NAME : " << pointer->canName[i] << endl;

cout << "YEAR : " << pointer->canYear[i] << endl;

cout << "FACULTY : " << pointer->canFaculty[i] << endl;

cout << "TARGETED POSITION : " << pointer->canTargetedPosition[i] << endl;

cout << "MANIFESTO : " << pointer->canManifesto[i] << endl;

if(!(i+1 == pointer->countCan))

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

}

cout << "-----------------------------------------\n\n" << endl;

}

void displayInfo(Queue\* pointer, int index){

cout << "\n---------------INFORMATION---------------" << endl;

cout << "NAME : " << pointer->getName(index) << endl;

cout << "IC : " << pointer->getIC(index) << endl;

cout << "MATRIC NUMBER : " << pointer->getMatricNo(index) << endl;

cout << "COURSE : " << pointer->getCourse(index) << endl;

cout << "AGE : " << pointer->getAge(index) << endl;

cout << "VOTE STATUS : ";

if(pointer->getVoteStatus(index))

cout << "DONE" << endl;

else

cout << "NOT YET" << endl;

cout << "-----------------------------------------\n\n" << endl;

}

int validity(Queue\* pointer, string icOrMatric){

// Declare new variable fc: first character to hold the first character of icOrMatric

char fc = icOrMatric.at(0);

int index = -1, out = 0, i = 0;

while(i < pointer->count){

if((fc == 'A') || (fc == 'B') || (fc == 'X')){

if(pointer->getMatricNo(i) == icOrMatric){

index = i;

if(!pointer->getVoteStatus(i))

break;

else

out = -2;

}

}

else{

if(pointer->getIC(i) == icOrMatric){

index = i;

if(!pointer->getVoteStatus(i))

break;

else

out = -2;

}

}

i++;

}

if (out == -2){

cout << "You already vote! Each person can vote only once!\n" << endl;

displayInfo(pointer, index);

index = -2;

}

return index;

}

Queue\* createQueue(){

Queue\* newQueue = new Queue();

newQueue->front = newQueue->rear = NULL;

return newQueue;

}

void enterQueue(Queue\* pointer, string icOrMatric){

int index = 0;

index = validity(pointer, icOrMatric);

if(index >= 0 && !findVoter(pointer, icOrMatric)){

QueueNode\* temp = registerVoter(pointer, index);

if(pointer->rear == NULL){

pointer->front = pointer->rear = temp;

}

else{

pointer->rear->next = temp;

pointer->rear = temp;

}

cout << "The user information is entering the Queue!" << endl;

displayInfo(pointer, index);

}

else if(findVoter(pointer, icOrMatric)){

cout << "You already registered in the Queue!\n" << endl;

displayQueueList(pointer);

}

else if(index == -1){

cout << "The IC or Matric Number entered does not exist in the database!\n" << endl;

}

}

QueueNode\* deleteQueue(Queue\* pointer){

if(pointer->front == NULL)

return NULL;

QueueNode\* temp = pointer->front;

delete(temp);

pointer->front = pointer->front->next;

if(pointer->front == NULL)

pointer->rear = NULL;

return temp;

}

void registerMenu(Queue\* ptrQ){

string icOrMatric;

cout << "<<<<<WELCOME TO OUR REGISTER USER>>>>>\n\n";

cout << "Enter Your Matric/ IC Number: ";

cin.ignore();

getline(cin, icOrMatric);

enterQueue(ptrQ, icOrMatric);

cout << "<<<<<<<<<END OF REGISTER USER>>>>>>>>>\n\n";

return;

}

void votingMenu(Queue\* ptrQ){

int can[4];

if(ptrQ->front == NULL){

cout << "The Queue is still empty! Please register first" << endl;

return;

}

cout << "<<<<<<<<WELCOME TO OUR VOTING MENU>>>>>>>>\n\n";

displayCandidateList(ptrQ);

cout << "You are need to vote for 4 candidates." << endl;

for(int i = 0; i < 4; i++){

cout << "Enter your chosen candidate number: ";

cin >> can[i];

if(can[i] < 1 || can[i] > ptrQ->countCan){

cout << "Invalid Vote! You need to re-vote again!" << endl;

system("PAUSE");

system("CLS");

votingMenu(ptrQ);

return;

}

}

for(int i = 0; i < 4; i++){

for(int j = 0; j < 4; j++){

if((can[i] == can[j]) && (i!=j)){

cout << "Invalid Vote! You need to re-vote again!"

<< "\nYou cannot vote a candidate more than once!"

<< "\nYou need re-vote again!" << endl;

system("PAUSE");

system("CLS");

votingMenu(ptrQ);

return;

}

}

}

for(int i = 0; i < 4; i++){

ptrQ->canVote[can[i]-1]++;

}

ptrQ->setVoteStatus(true);

deleteQueue(ptrQ);

cout << "<<<<<<<<<<<<END OF VOTING MENU>>>>>>>>>>>>\n\n";

}

void menu(Queue \*ptrQ){

int option;

cout << "<<<<<<<WELCOME TO OUR VOTING SYSTEM>>>>>>>\n\n";

cout << "1. Register User" << endl;

cout << "2. Voting" << endl;

cout << "3. List Queue" << endl;

cout << "4. Exit" << endl;

cout << "\nOption: ";

cin >> option;

cout << "<<<<<<<<<<<END OF VOTING SYSTEM>>>>>>>>>>>\n\n";

switch(option){

case 1:

registerMenu(ptrQ);

break;

case 2:

votingMenu(ptrQ);

break;

case 3:

displayQueueList(ptrQ);

break;

case 4:

cout << "Thank you for using our Voting System..." << endl;

system("PAUSE");

return;

break;

default:

cout << "ERROR! Invalid Input" << endl;

break;

}

system("PAUSE");

system("CLS");

menu(ptrQ);

}

int main(){

Queue\* ptrQ = createQueue();

menu(ptrQ);

ptrQ->updateFile();

return 0;

}