TARUMT
TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

BACS1014

Problem Solving And Programming

Assignment
2022/2023

Programme	:	RDS
Tutorial Group	:	3
Date Submitted to Tutor	:	30/12/2022

Team Members:

No	Student Name	Student ID
1.	Yam Jason	22WMR13662
2.	Wong Yee En	22WMR13659
3.	Tang Wen Zhi	22WMR13640

Declaration of Originality

I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own work. I understand that I will be penalized if I have not complied with TAR UMT's Plagiarism policy.

Signature: YEE EN Name: Wong Yee En Date: 31/12/2022

Declaration of Originality

I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own work. I understand that I will be penalized if I have not complied with TAR UMT's Plagiarism policy.

Signature: JASONName: Yam Jason Date: 31/12/2022

Declaration of Originality

I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own work. I understand that I will be penalized if I have not complied with TAR UMT's Plagiarism policy.

Signature: WEN ZHI Name: Tang Wen Zhi Date: 31/12/2022

No.	Team Member	Task(s) Allocated	Overall Contribution (%)
1.	Yam Jason	Programming, Flowchart, Program Testing & Outputs, Additional features.	35
2.	Wong Yee En	Programming, Flowchart, Additional features, Constants & variables	35
3.	Tang Wen Zhi	Programming, Flowchart, Purpose of the program	30

Coursework Declaration

We confirm that we have read and shall comply with all the terms and conditions of TAR University College's plagiarism policy.

We declare that this assignment is free from all forms of plagiarism and for all intents and purposes is our own properly derived work.

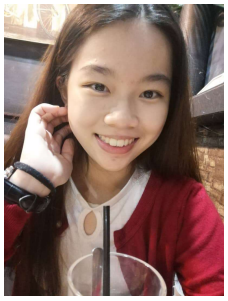
Signature	:	*Jason*	*Yee En*	*Wen Zhi*
Name	:	Yam Jason	Wong Yee En	Tang Wen Zhi
Photo	:			
Date	:	30/12/2022	30/12/2022	30/12/2022

Table of Content

Contents

Brief Description / Purpose of the program

In the college or the university, every study programme must go through the routine process of electing a programme representative. The purpose of the routine process is used to choose who they want to become a leader. The leader will act as an intermediary between the university's top administrators and the students' professors.

Therefore, the purpose of this program is to make convenience by speeding up the process of the election, so all the students can choose their favourite leader who will be Programme Representative via this electronic voting system.

Due to the old version of the election of voting, so as IT students we are responsible to modernize or automate this election through making a program coding to convert it into an electronic voting system which does not need manpower to work on it.

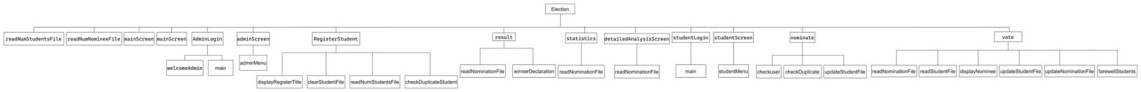
This program election will allow all the students to vote by using simple clicks via the devices. They can vote anywhere and anytime which is allowed. This electronic voting system will have two parts which are the users (voters) level and the election administrator level.

User part allows all the students to login, nominate and vote. The user will be limited to some parts which are fulfilled by the electronic voting criteria. For example, the user level only can accept 5 unique nominees as the candidates nominated for the election. With these effective process or function, the electronic voting can be carry on smoothly, automatically, and good.

Apart from this, the administrator level is to allow the administrator to register students, display the final result of the electronic voting, show the simple statistics, and some detailed analysis such as the percentages of votes for each nominee, number of female and male votes which have participated in this voting system. This process of this system is running digitally and automatically which can replace the old version of the old election voting system.

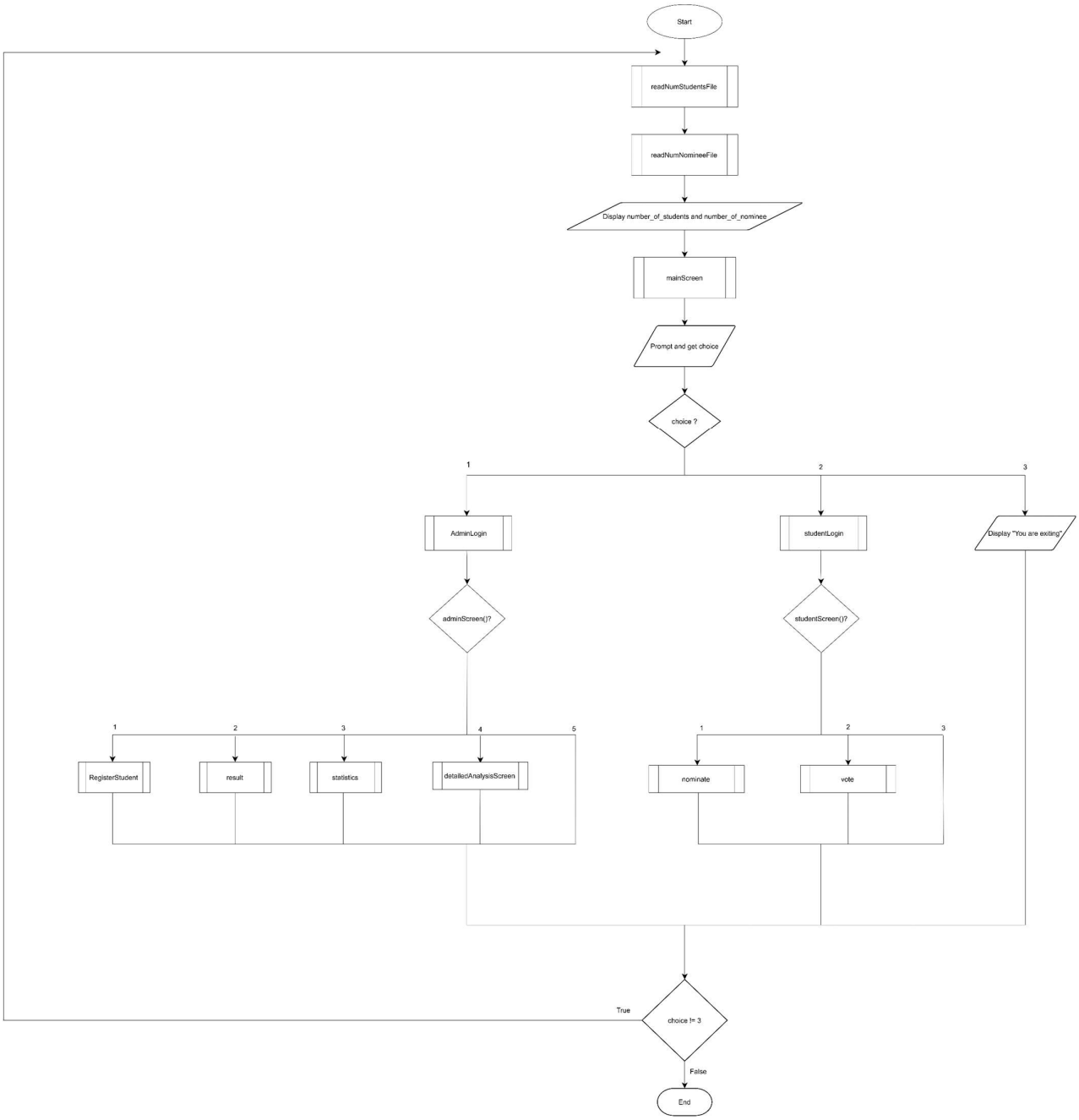
Overall Program Design

Structure Chart

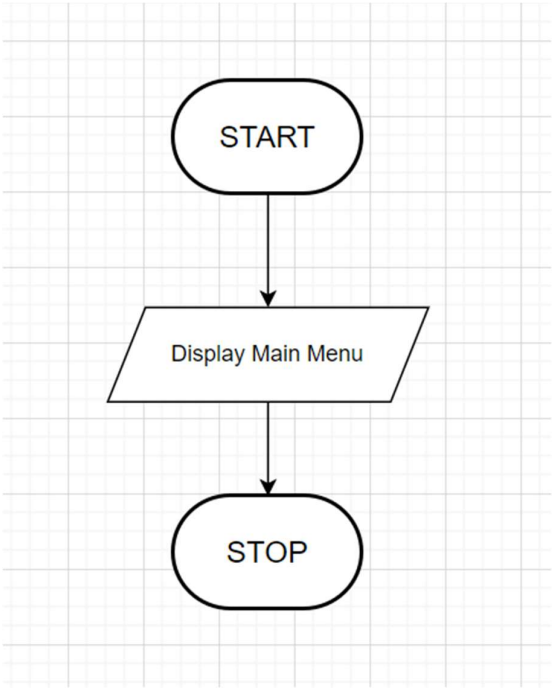


Flowchart

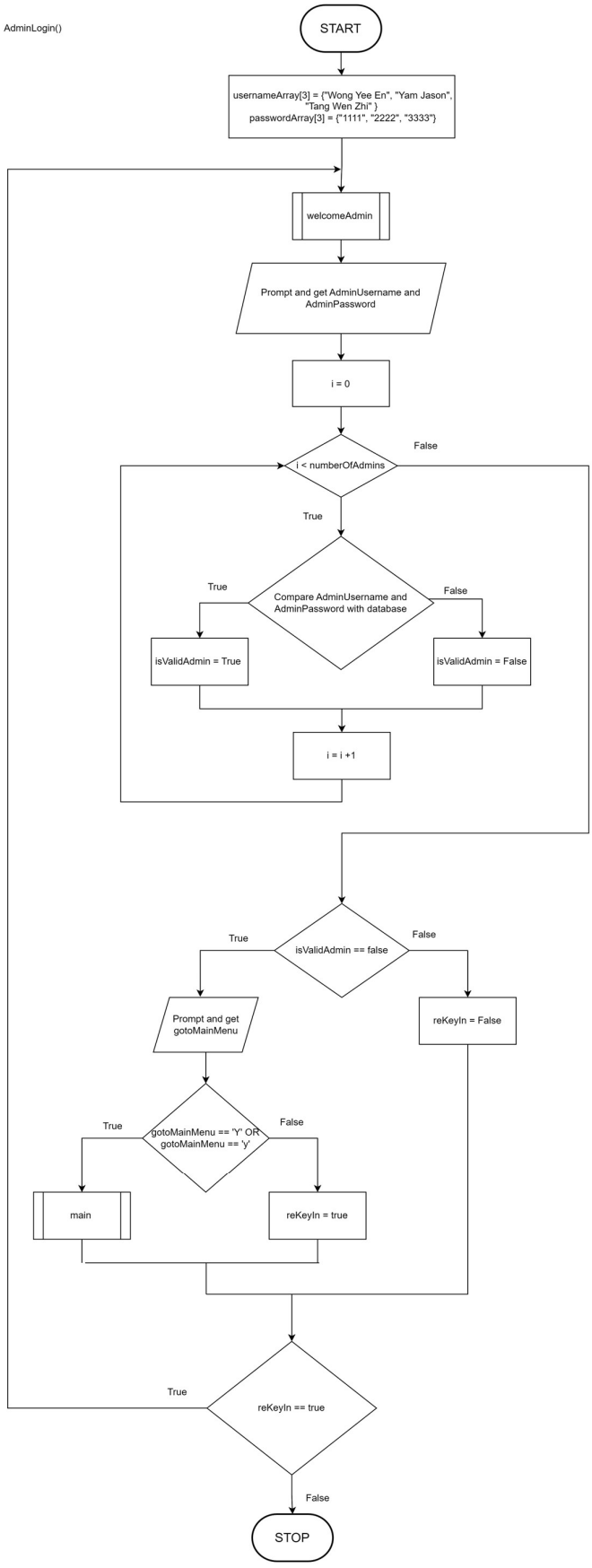
Main



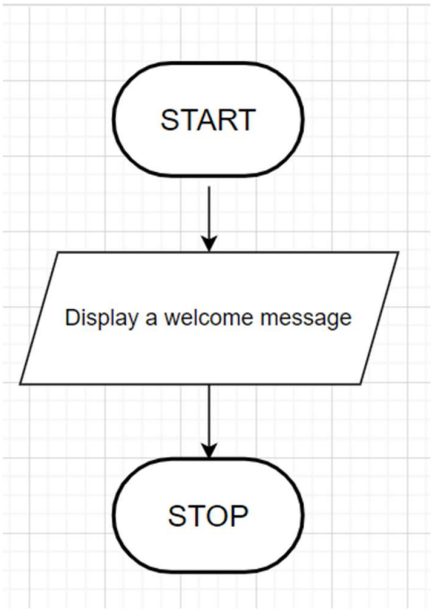
mainScreen()



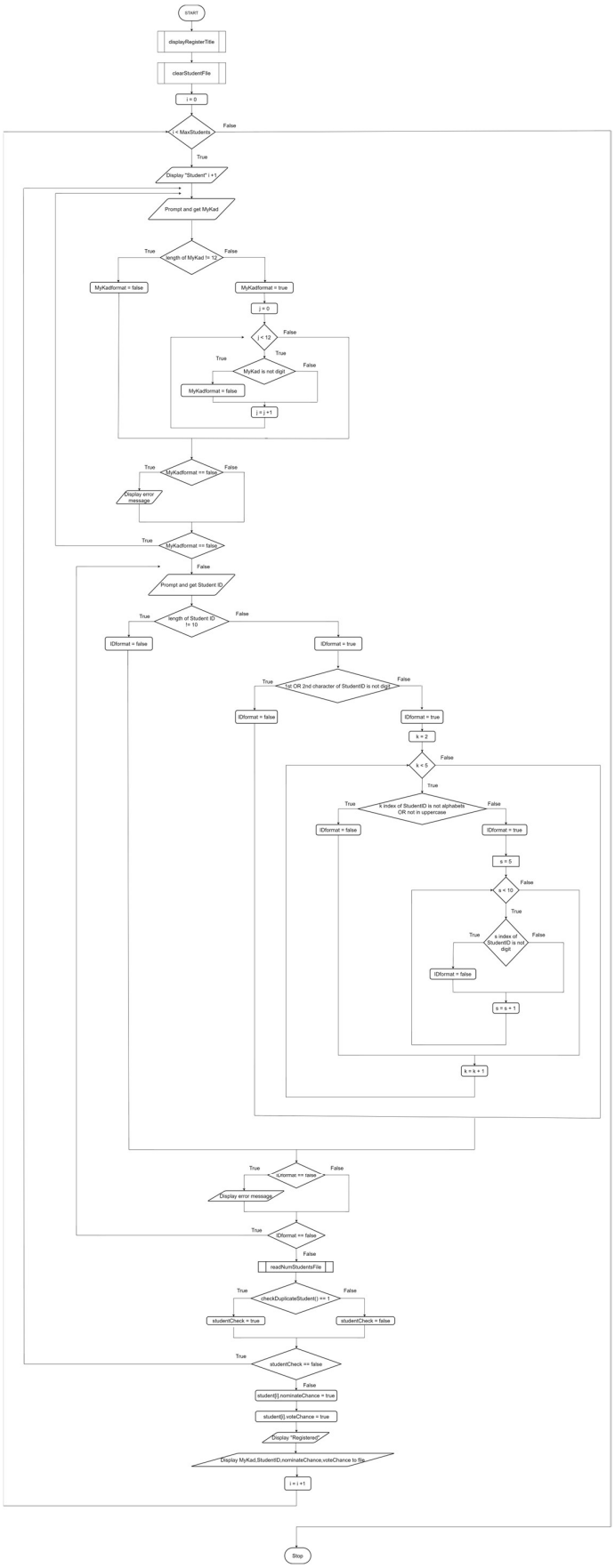
AdminLogin()


AdminLogin()
START
usernameArray[3] = {"Wong Yee En", "Yam Jason", "Tang Wen Zhi" }
passwordArray[3] = {"1111", "2222", "3333"}
welcomeAdmin
Prompt and get AdminUsername and AdminPassword
i = 0
i < numberOfAdmins
False
True
Compare AdminUsername and AdminPassword with database
True
False
isValidAdmin = True
isValidAdmin = False
i = i +1
isValidAdmin == false
True
False
Prompt and get gotoMainMenu
reKeyIn = False
gotoMainMenu == 'Y' OR gotoMainMenu == 'y'
True
False
main
reKeyIn = true
reKeyIn == true
True
False
STOP

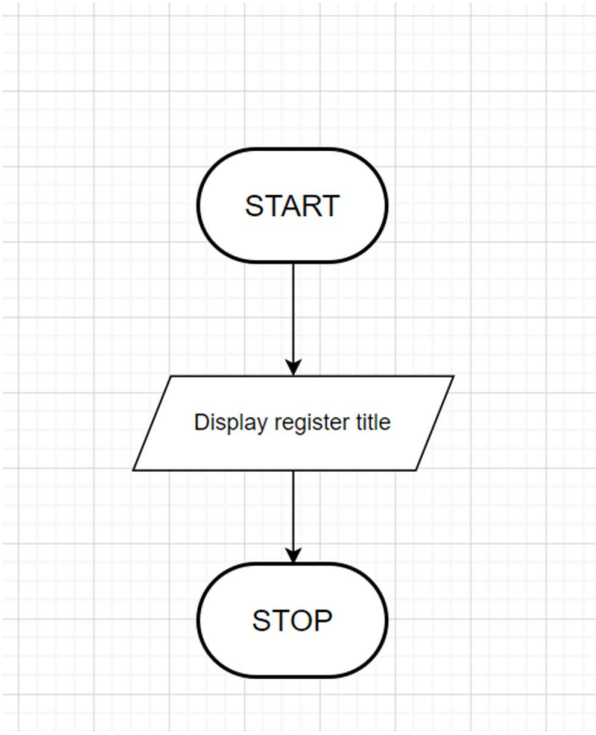
welcomeAdmin()



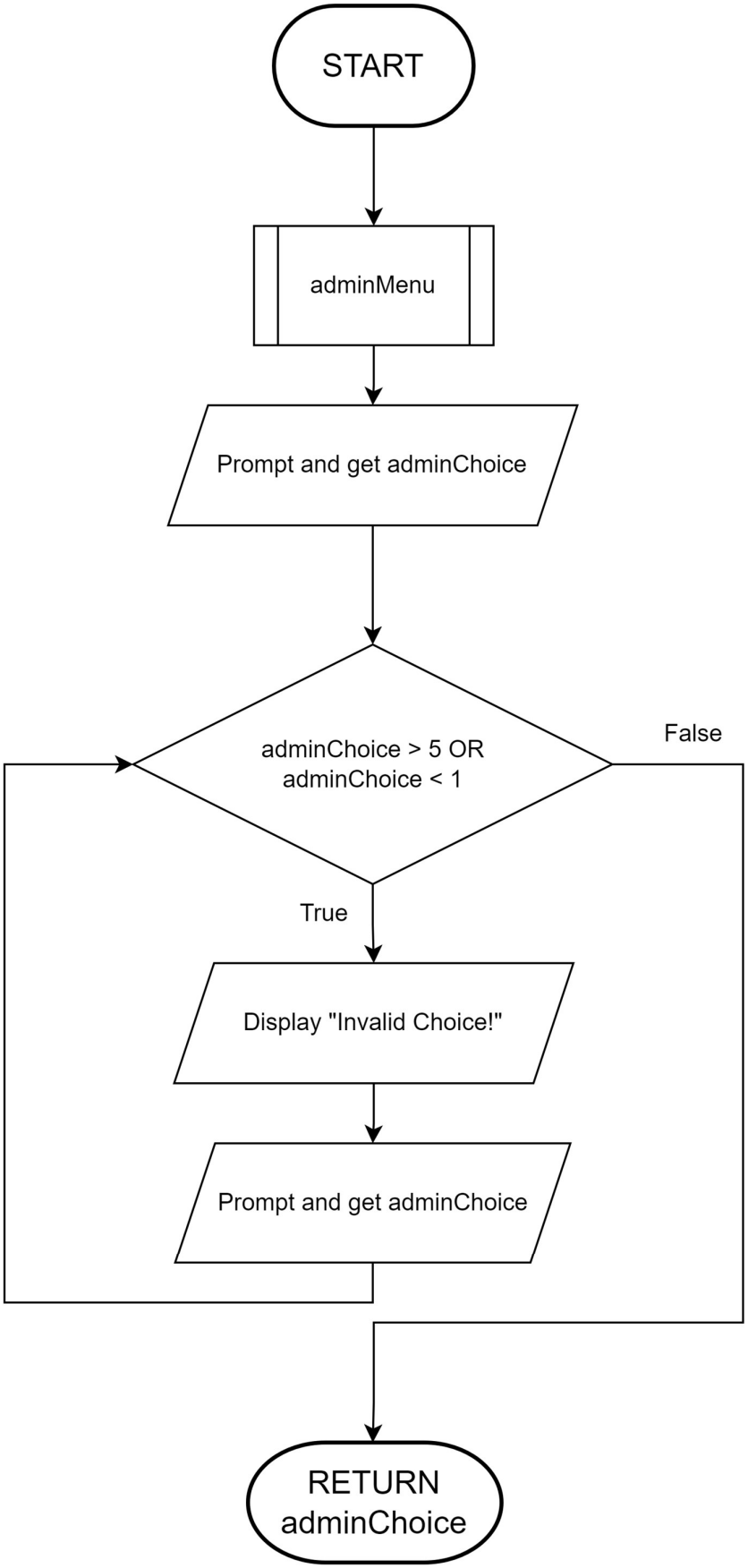
RegisterStudent()



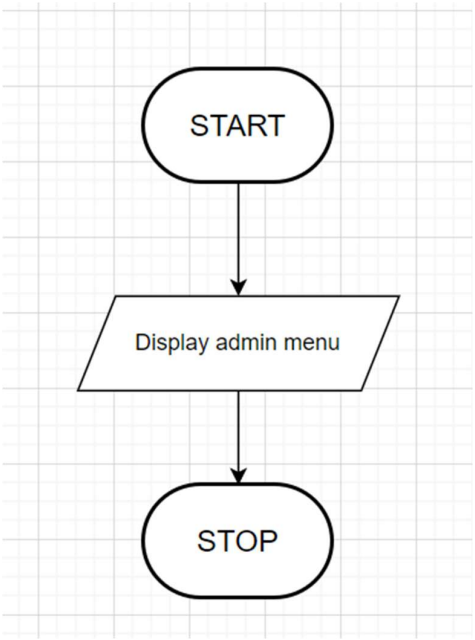
displayRegisterTitle()



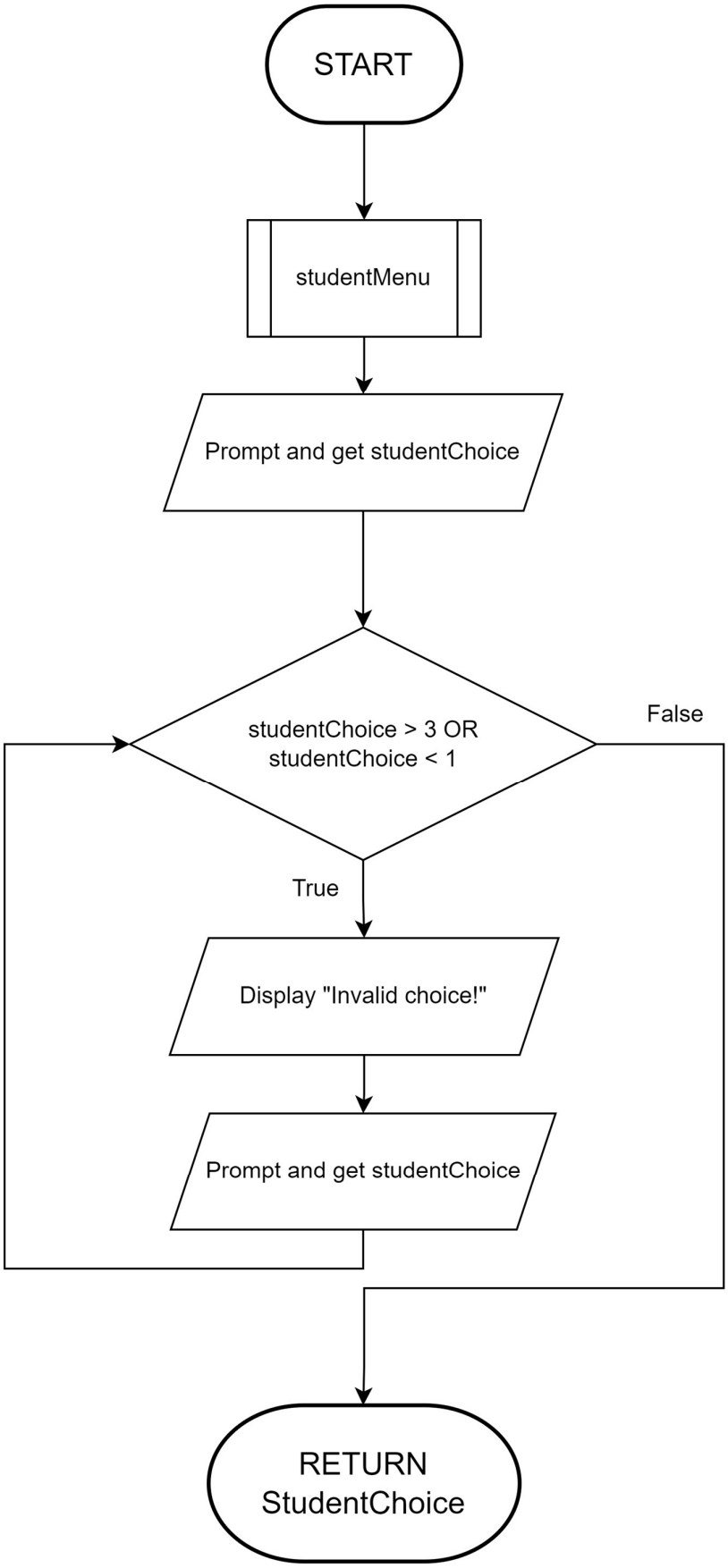
adminScreen()



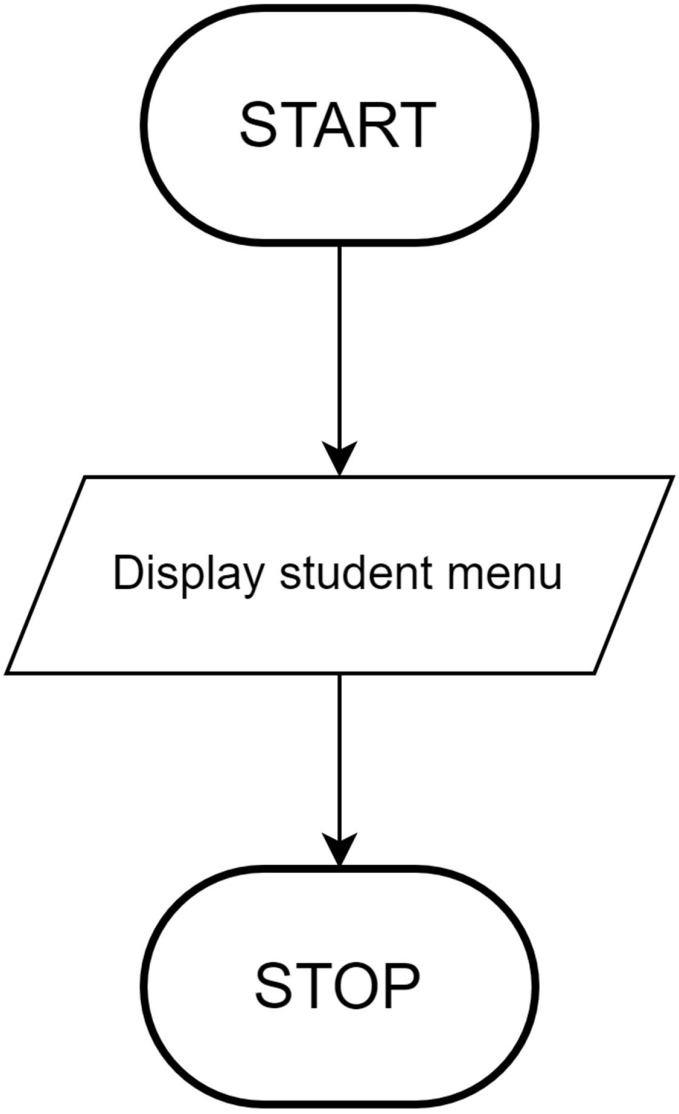
adminMenu()



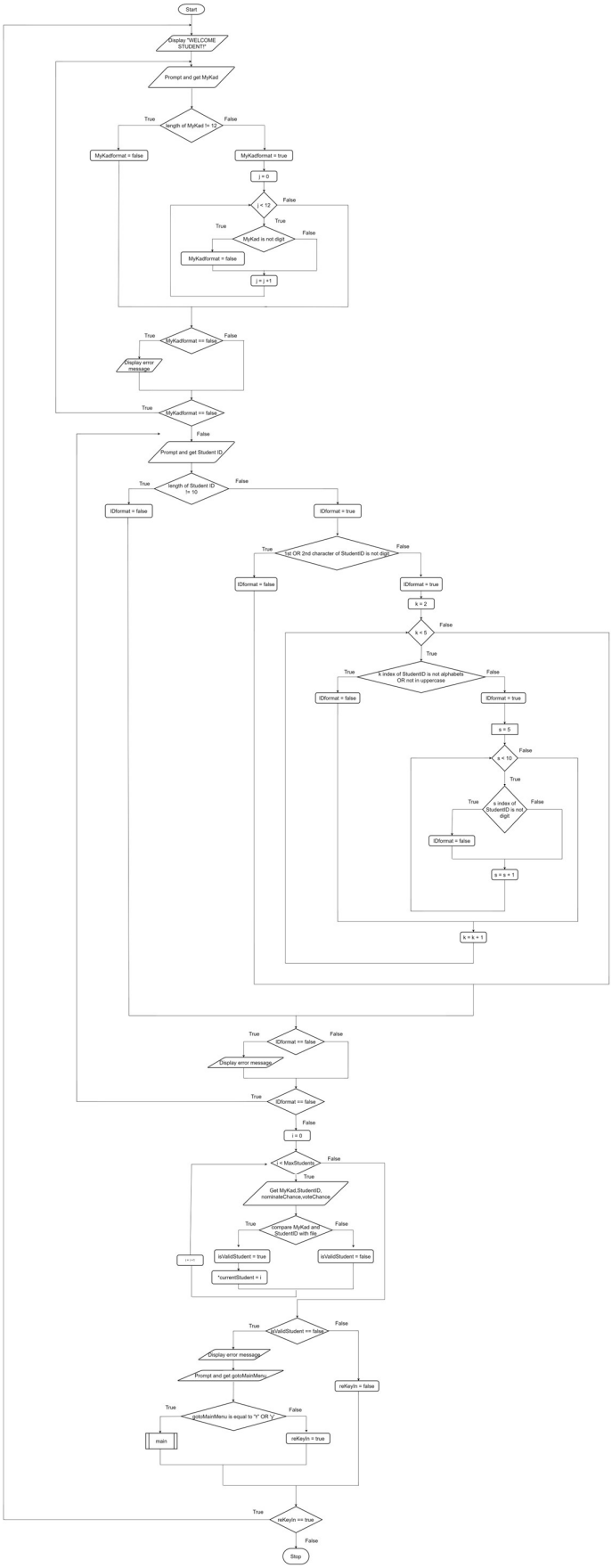
studentScreen()



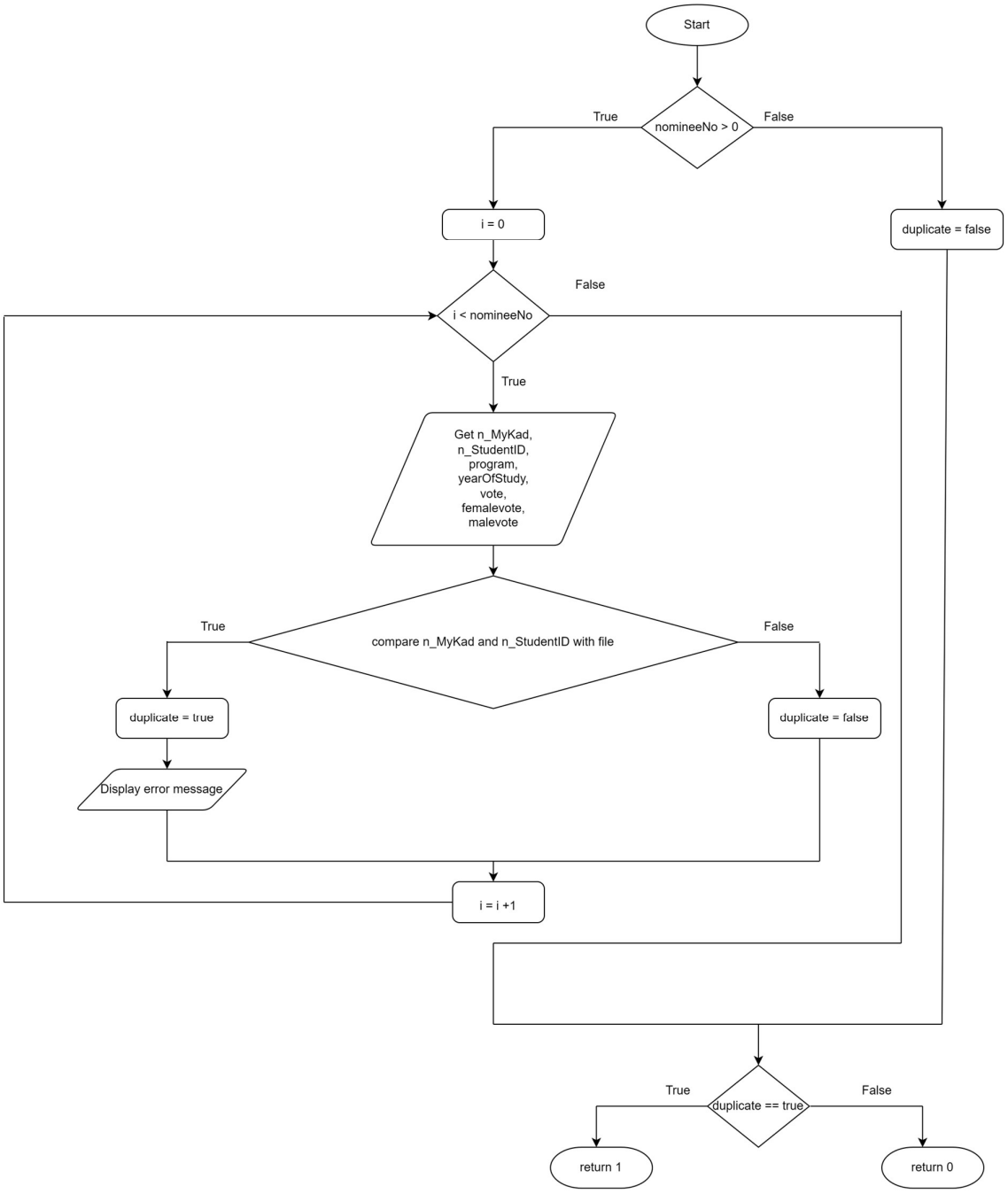
studentMenu()



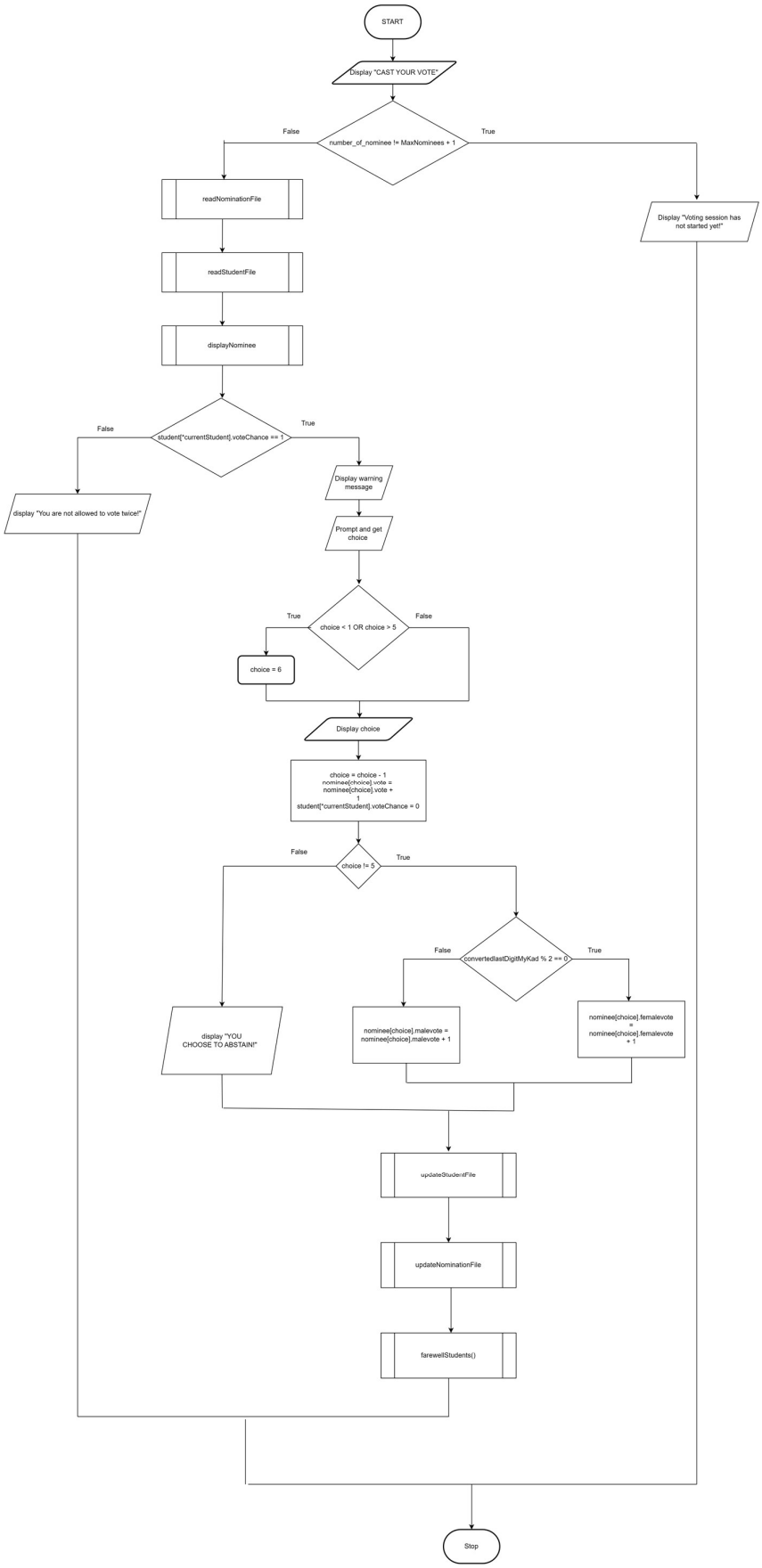
studentLogin()



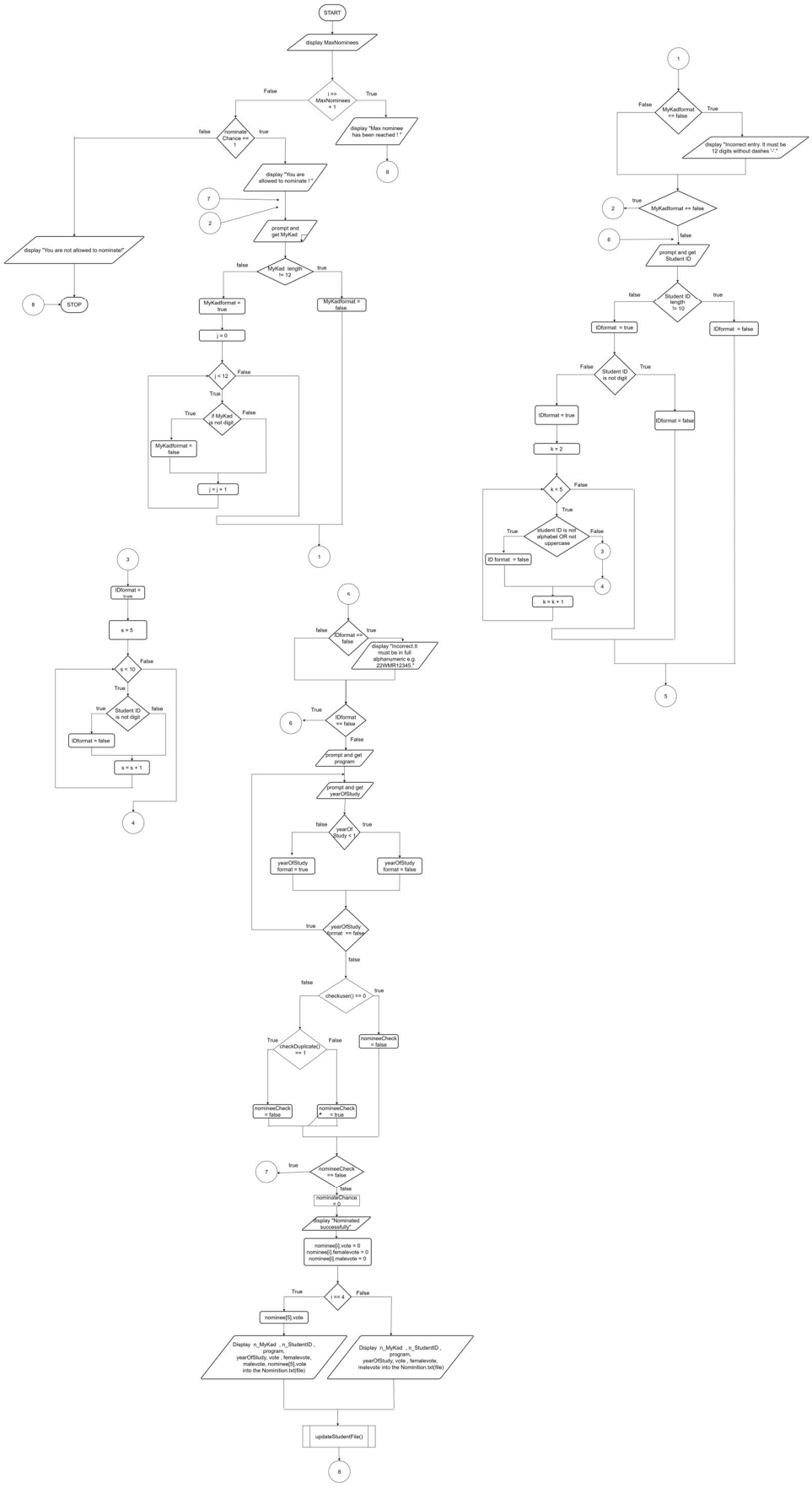
checkDuplicate()



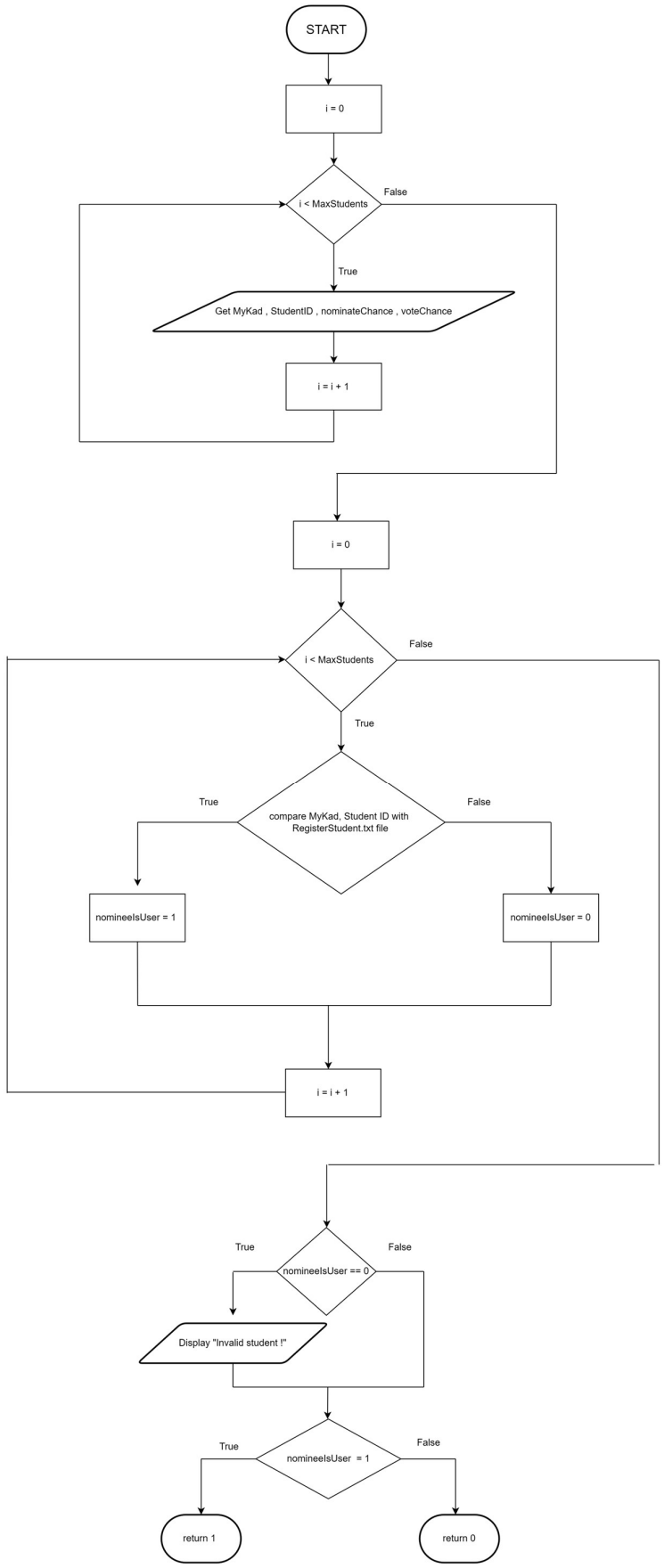
vote()


START
Display "CAST YOUR VOTE"
number_of_nominee != MaxNominees + 1
False
True
Display "Voting session has not started yet!"
readNominationFile
readStudentFile
displayNominee
student[*currentStudent].voteChance == 1
False
True
display "You are not allowed to vote twice!"
Display warning message
Prompt and get choice
choice < 1 OR choice > 5
True
False
choice = 6
Display choice
choice = choice - 1
nominee[choice].vote = nominee[choice].vote + 1
student[*currentStudent].voteChance = 0
choice != 5
False
True
convertedlastDigitMyKad % 2 == 0
False
True
display "YOU CHOOSE TO ABSTAIN!"
nominee[choice].malevote = nominee[choice].malevote + 1
nominee[choice].femalevote = nominee[choice].femalevote + 1
updateStudentFile
updateNominationFile
farewellStudents()
Stop

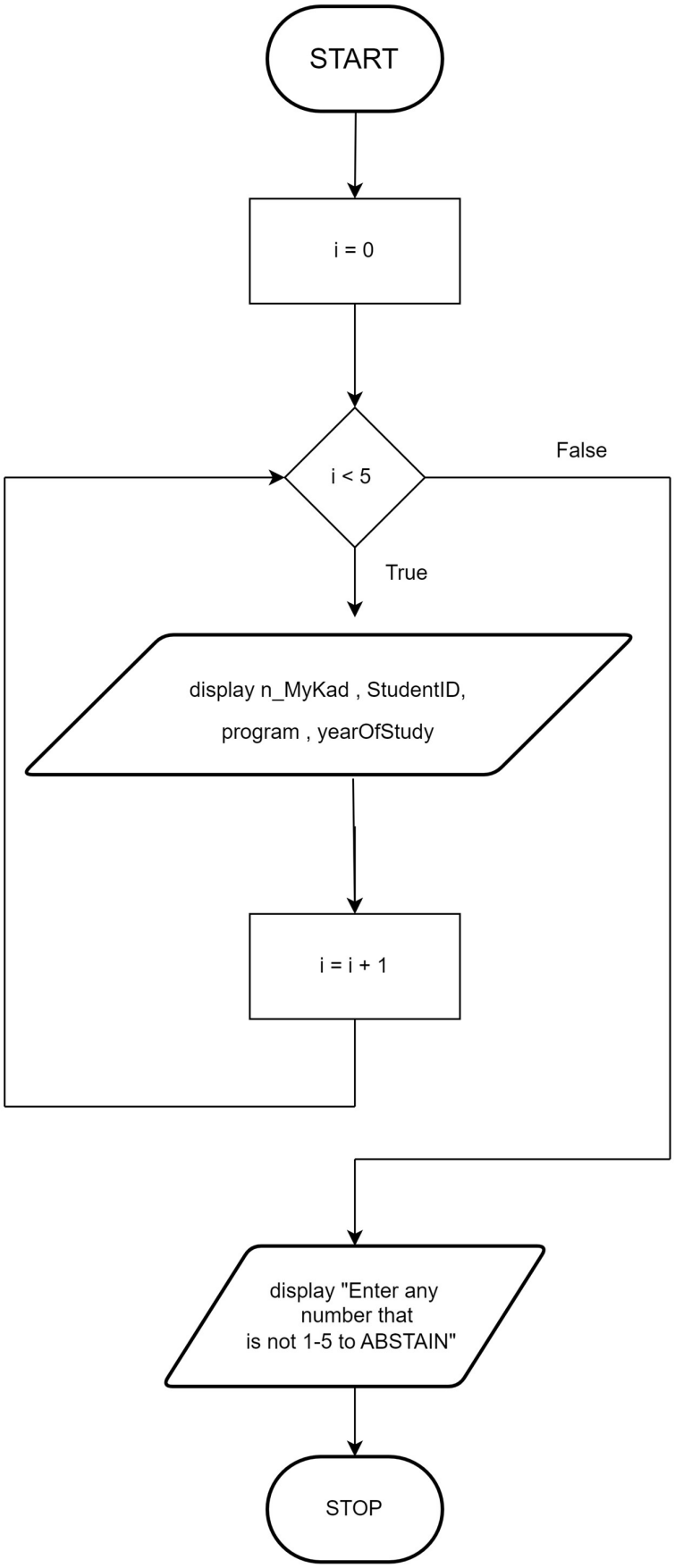
nominate()


START
display MaxNominees
i == MaxNominees + 1
False
True
display "Max nominee has been reached ! "
8
nominate Chance == 1
false
true
display "You are allowed to nominate ! "
7
2
prompt and get MyKad
display "You are not allowed to nominate!"
8
STOP
MyKad length != 12
false
true
MyKadformat = true
MyKadformat = false
j = 0
j < 12
False
True
if MyKad is not digit
True
False
MyKadformat = false
j = j + 1
1
MyKadformat == false
False
True
display "Incorrect entry. It must be 12 digits without dashes '-' "
2
true
MyKadformat == false
6
false
prompt and get Student ID
Student ID length != 10
false
true
IDformat = true
IDformat = false
Student ID is not digit
False
True
IDformat = true
IDformat = false
k = 2
k < 5
False
True
Student ID is not alphabet OR not uppercase
True
False
ID format = false
3
4
k = k + 1
5
3
IDformat = true
s = 5
s < 10
False
True
Student ID is not digit
true
false
IDformat = false
s = s + 1
4
4
IDformat == false
false
true
display "Incorrect.It must be in full alphanumeric e.g. 22WMR12345."
6
IDformat == false
True
prompt and get program
prompt and get yearOfStudy
yearOf Study < 1
false
true
yearOfStudy format = true
yearOfStudy format = false
yearOfStudy format. == false
true
false
checkuser() == 0
false
true
checkDuplicate() == 1
True
False
nomineeCheck = false
nomineeCheck = false
nomineeCheck = true
nomineeCheck == false
7
true
false
nominateChance = 0
display "Nominated successfully"
nominee[].vote = 0
nominee[].femalevote = 0
nominee[].malevote = 0
i == 4
True
False
nominee[5].vote
Display n_MyKad , n_StudentID , program, yearOfStudy, vote , femalevote, malevote, nominee[5].vote into the Nomnition.txt(file)
Display n_MyKad , n_StudentID , program, yearOfStudy, vote , femalevote, malevote into the Nomnition.txt(file)
updateStudentFile()
8

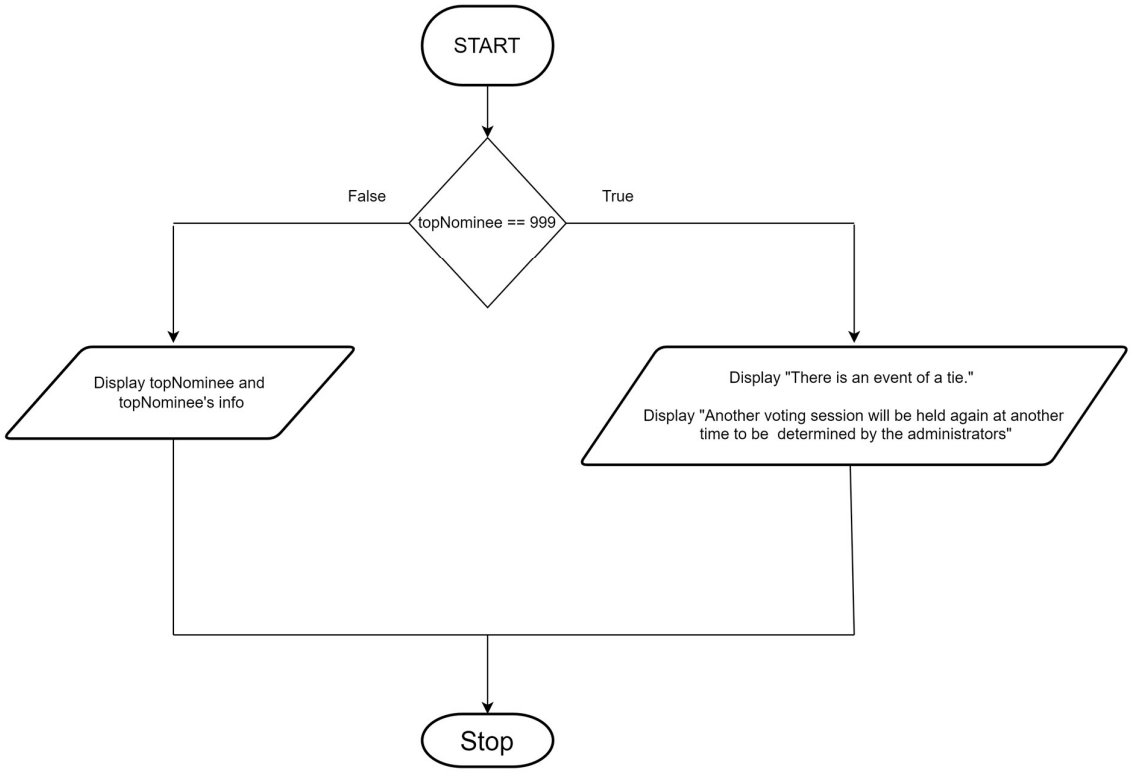
checkuser()


START
i = 0
i < MaxStudents
False
True
Get MyKad , StudentID , nominateChance , voteChance
i = i + 1
i = 0
i < MaxStudents
False
True
True
compare MyKad, Student ID with RegisterStudent.txt file
False
nomineeIsUser = 1
nomineeIsUser = 0
i = i + 1
True
nomineeIsUser == 0
False
Display "Invalid student !"
True
nomineeIsUser = 1
False
return 1
return 0

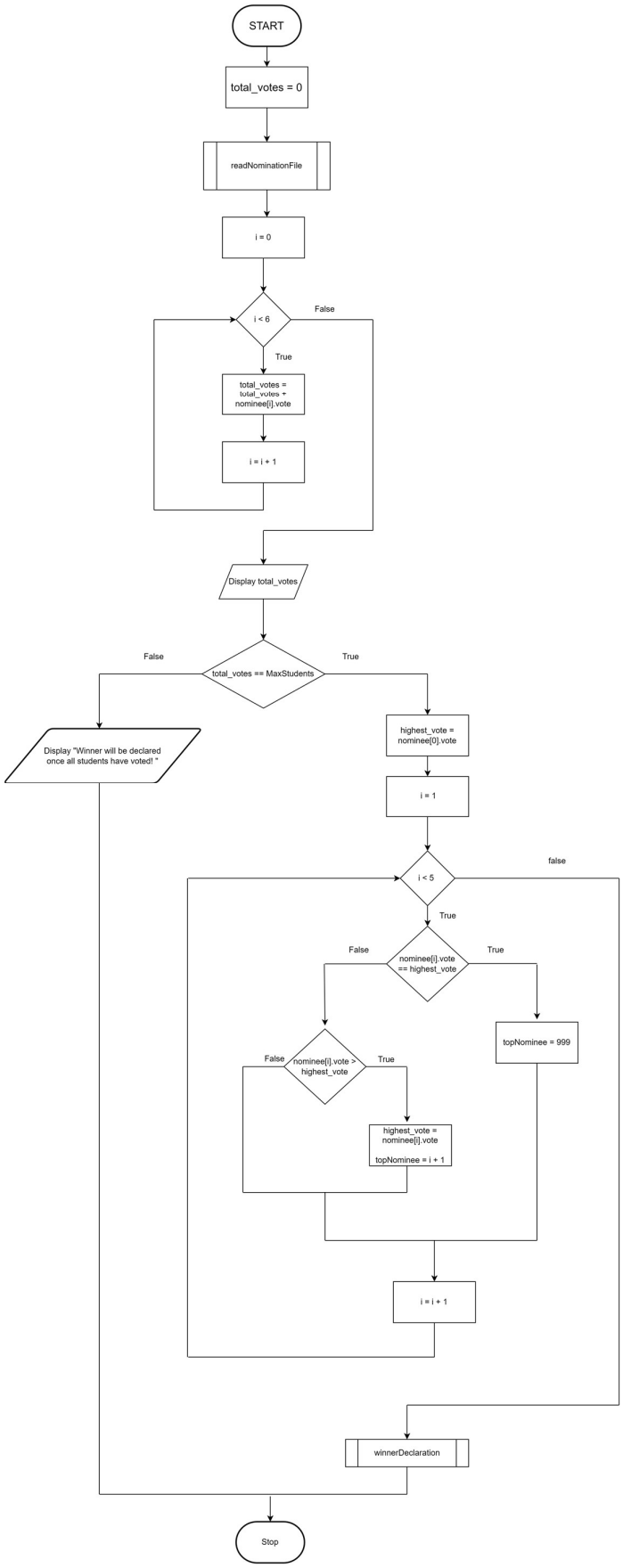
displayNominee()



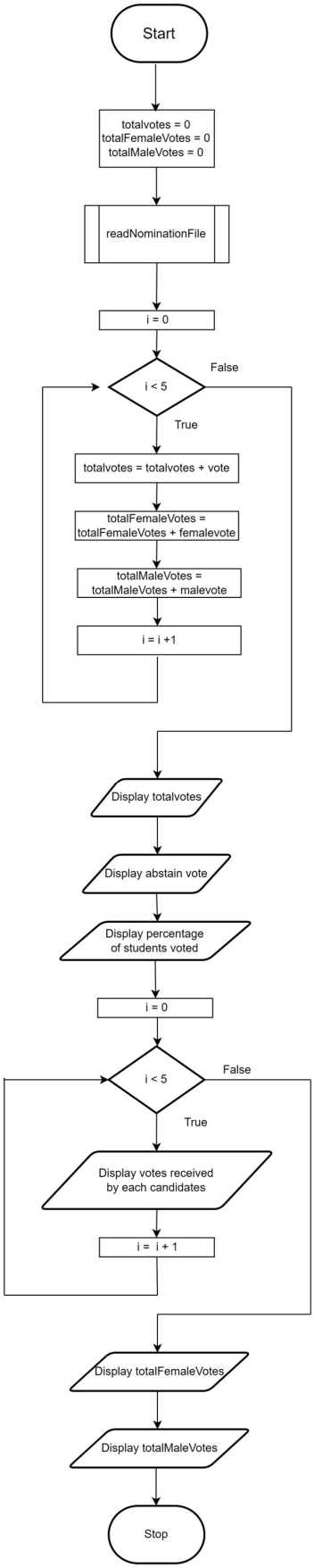
winnerDeclaraion()



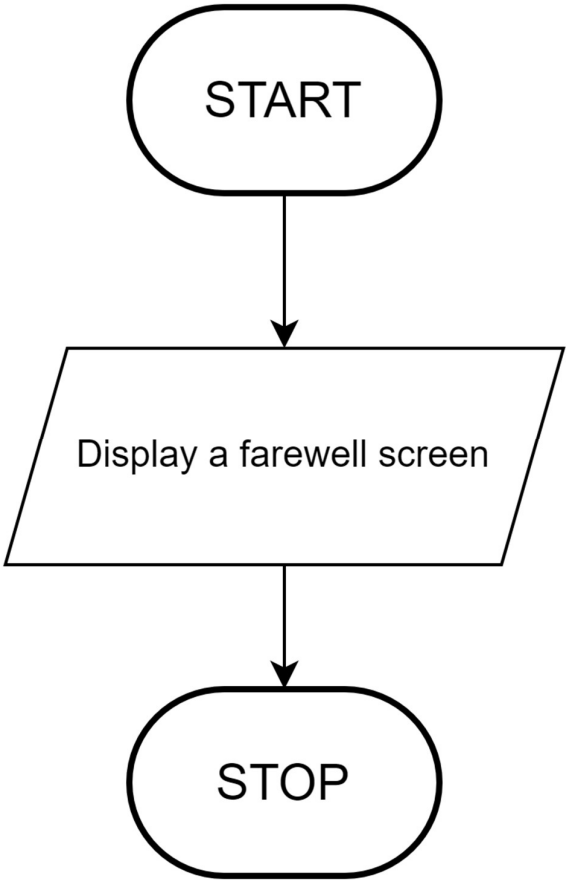
result()


START
total_votes = 0
readNominationFile
i = 0
i < 6
False
True
total_votes = total_votes + nominee[i].vote
i = i + 1
Display total_votes
False
total_votes == MaxStudents
True
Display "Winner will be declared once all students have voted! "
highest_vote = nominee[0].vote
i = 1
i < 5
false
True
False
nominee[i].vote == highest_vote
True
topNominee = 999
False
nominee[i].vote > highest_vote
True
highest_vote = nominee[i].vote
topNominee = i + 1
i = i + 1
winnerDeclaration
Stop

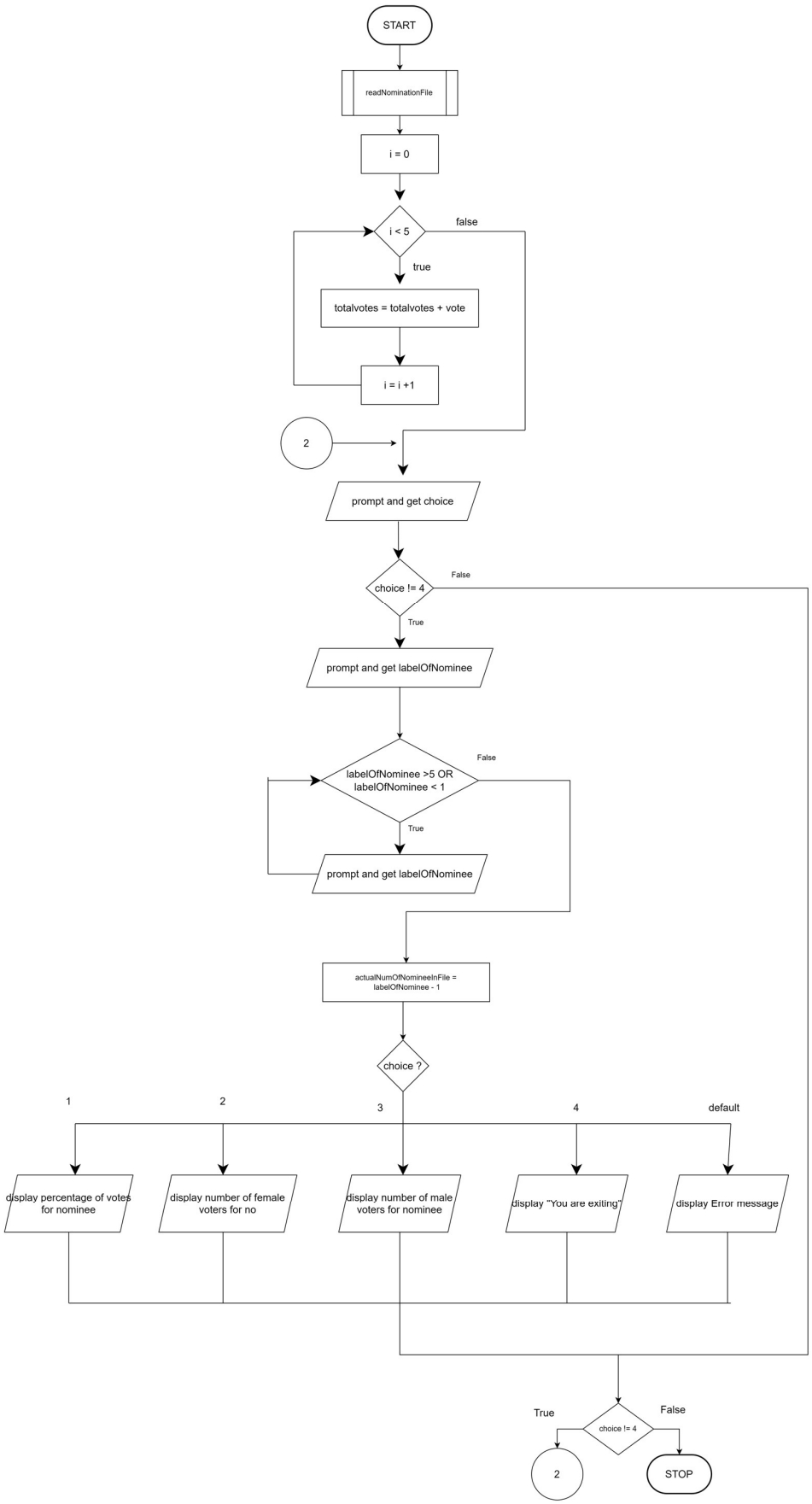
statistics()


Start
totalvotes = 0
totalFemaleVotes = 0
totalMaleVotes = 0
readNominationFile
i = 0
i < 5
False
True
totalvotes = totalvotes + vote
totalFemaleVotes = totalFemaleVotes + femalevote
totalMaleVotes = totalMaleVotes + malevote
i = i +1
Display totalvotes
Display abstain vote
Display percentage of students voted
i = 0
i < 5
False
True
Display votes received by each candidates
i = i + 1
Display totalFemaleVotes
Display totalMaleVotes
Stop

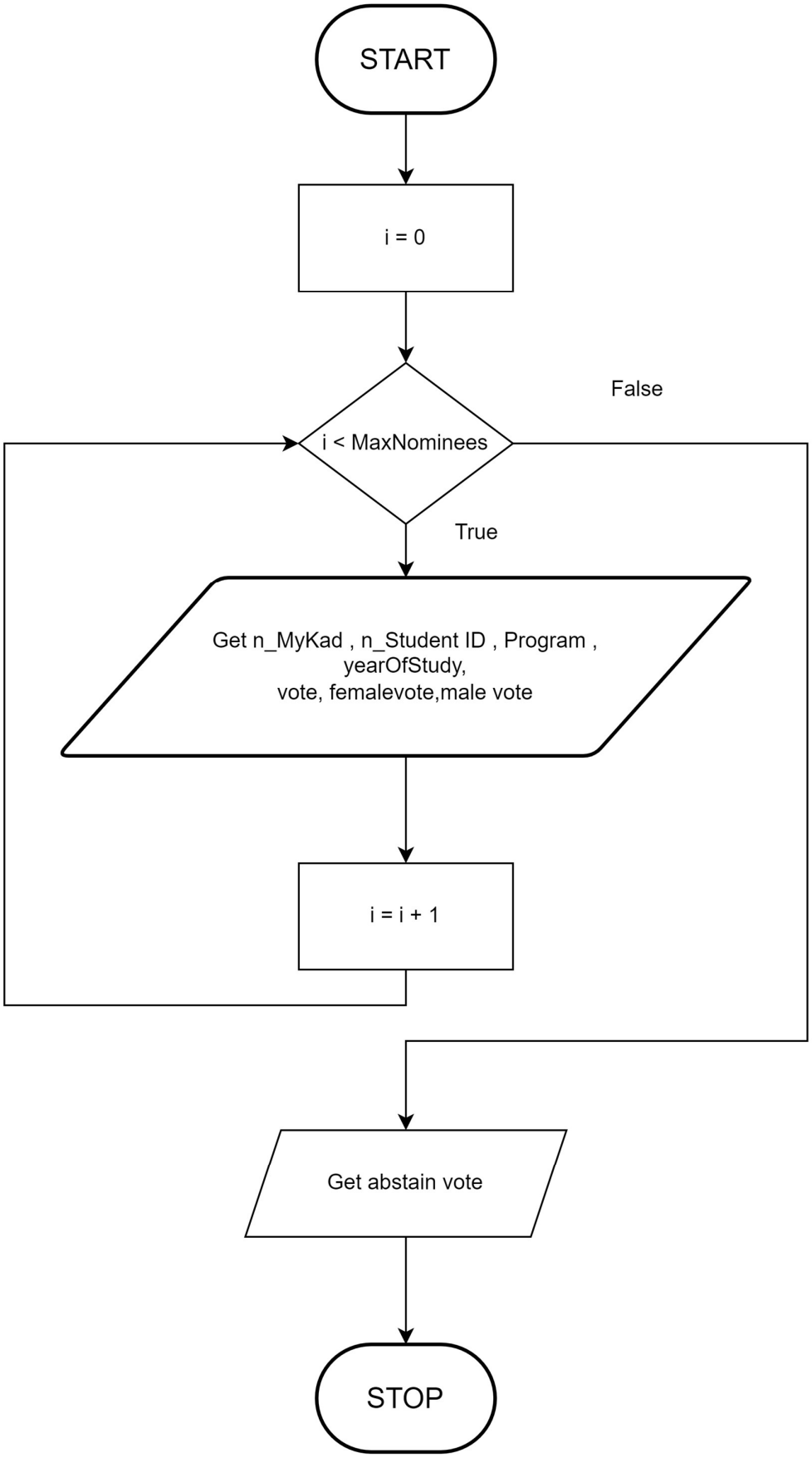
farewellStudents()



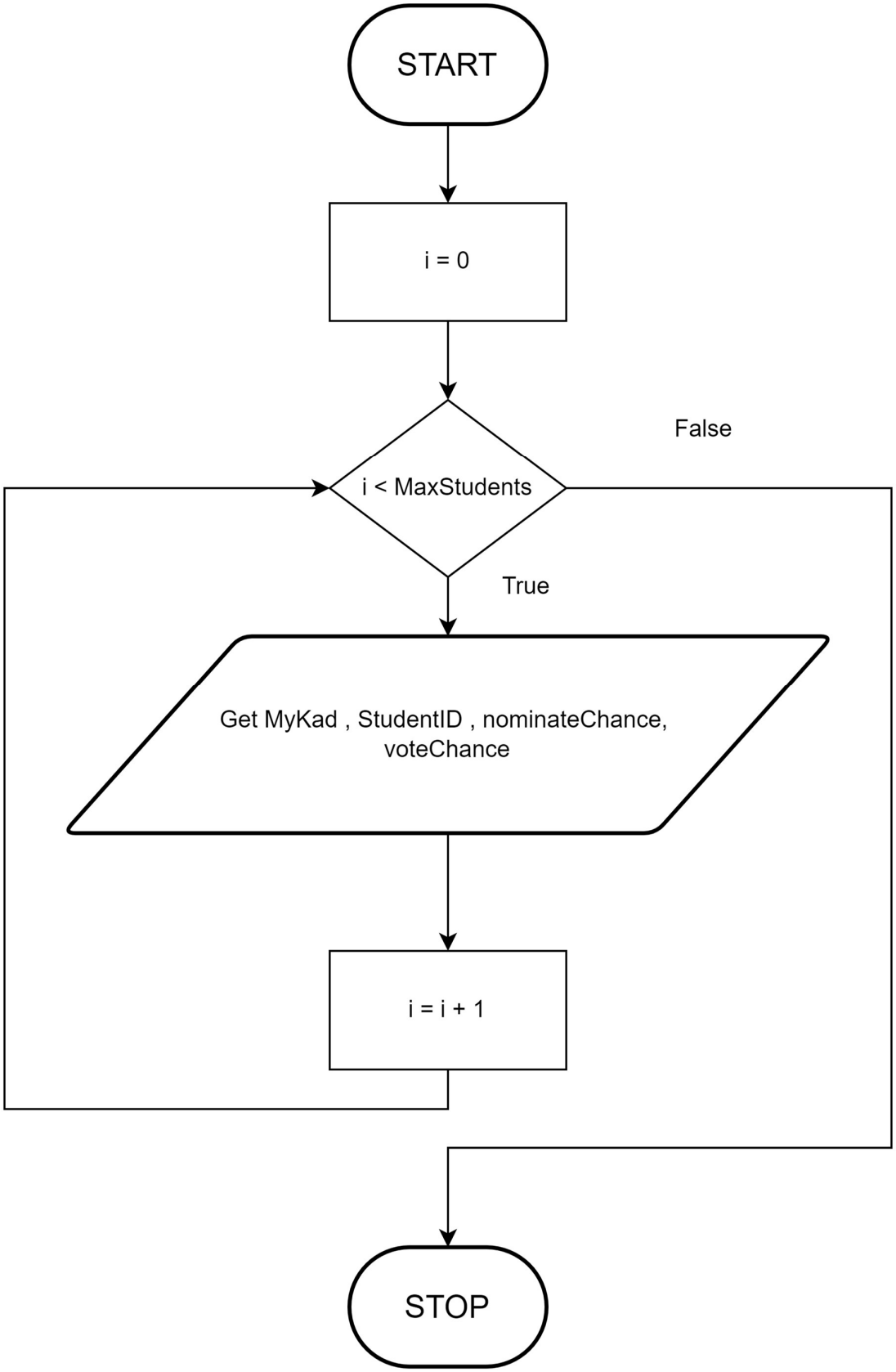
detailedAnalysisScreen()



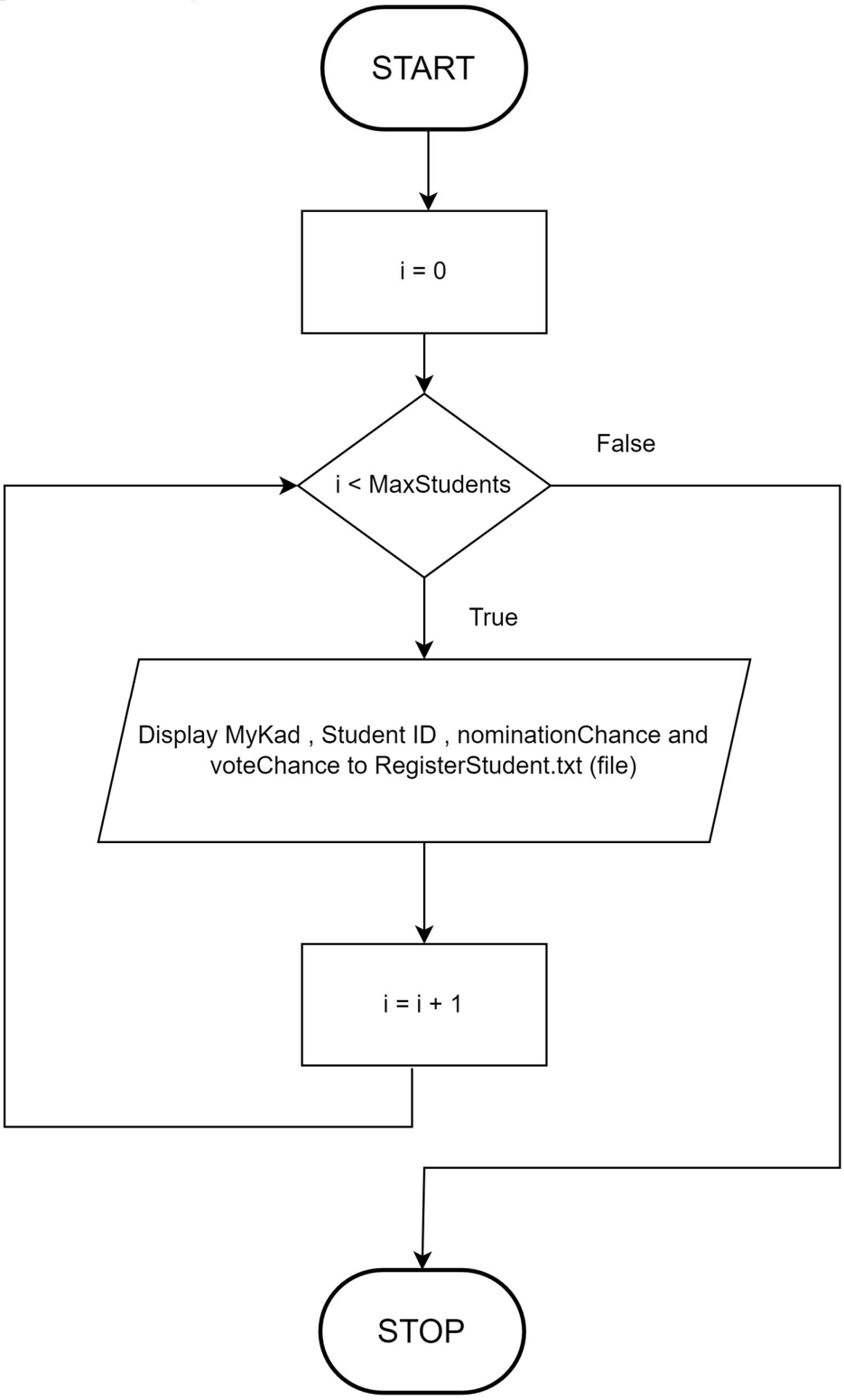
readNominationFile()



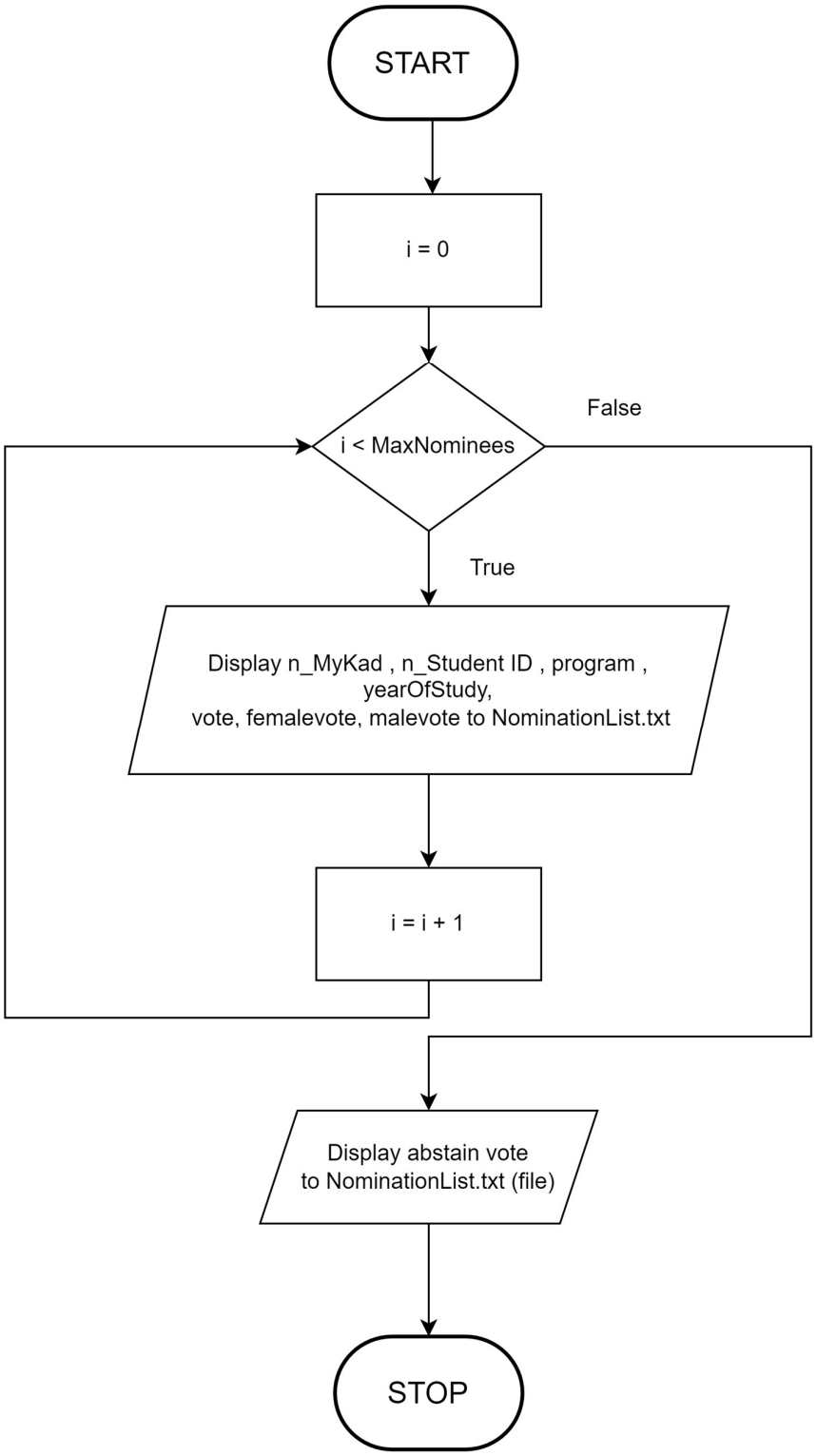
readStudentFile()



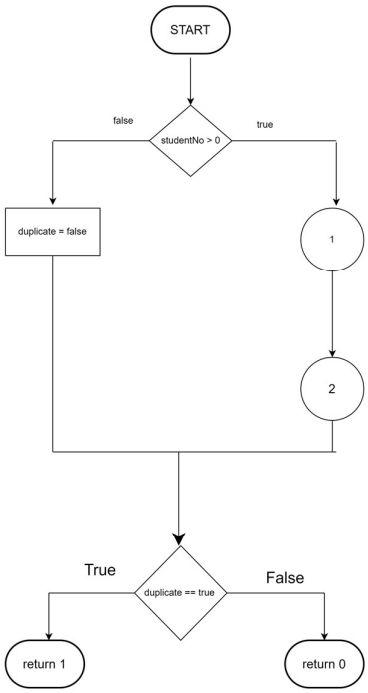
updateStudentFile()

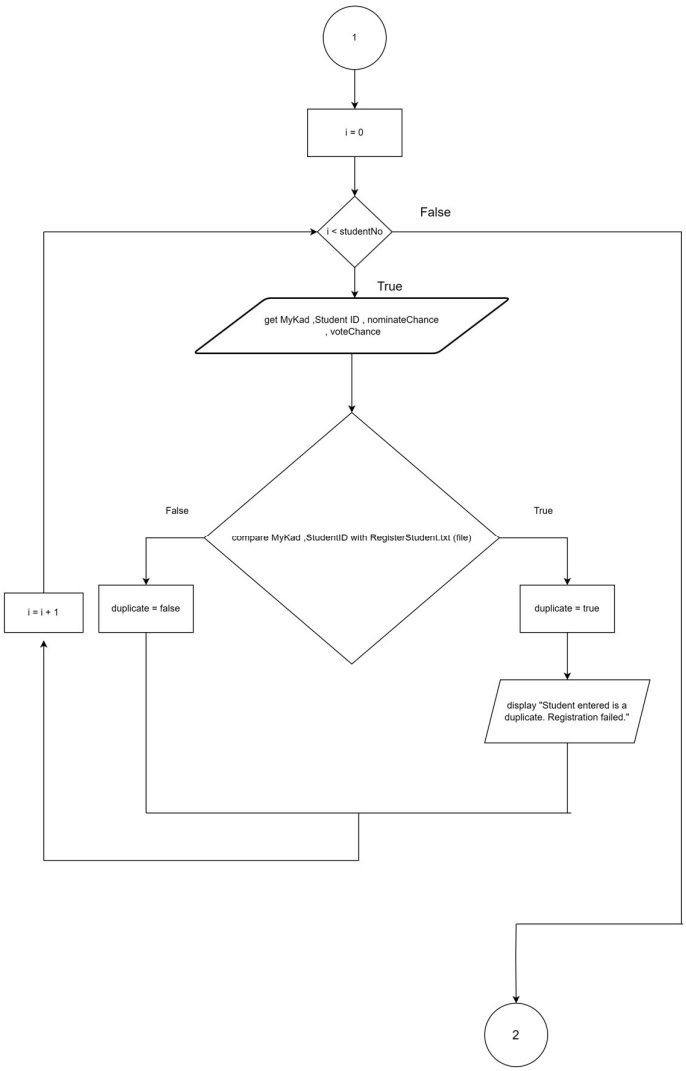


updateNominationFile()

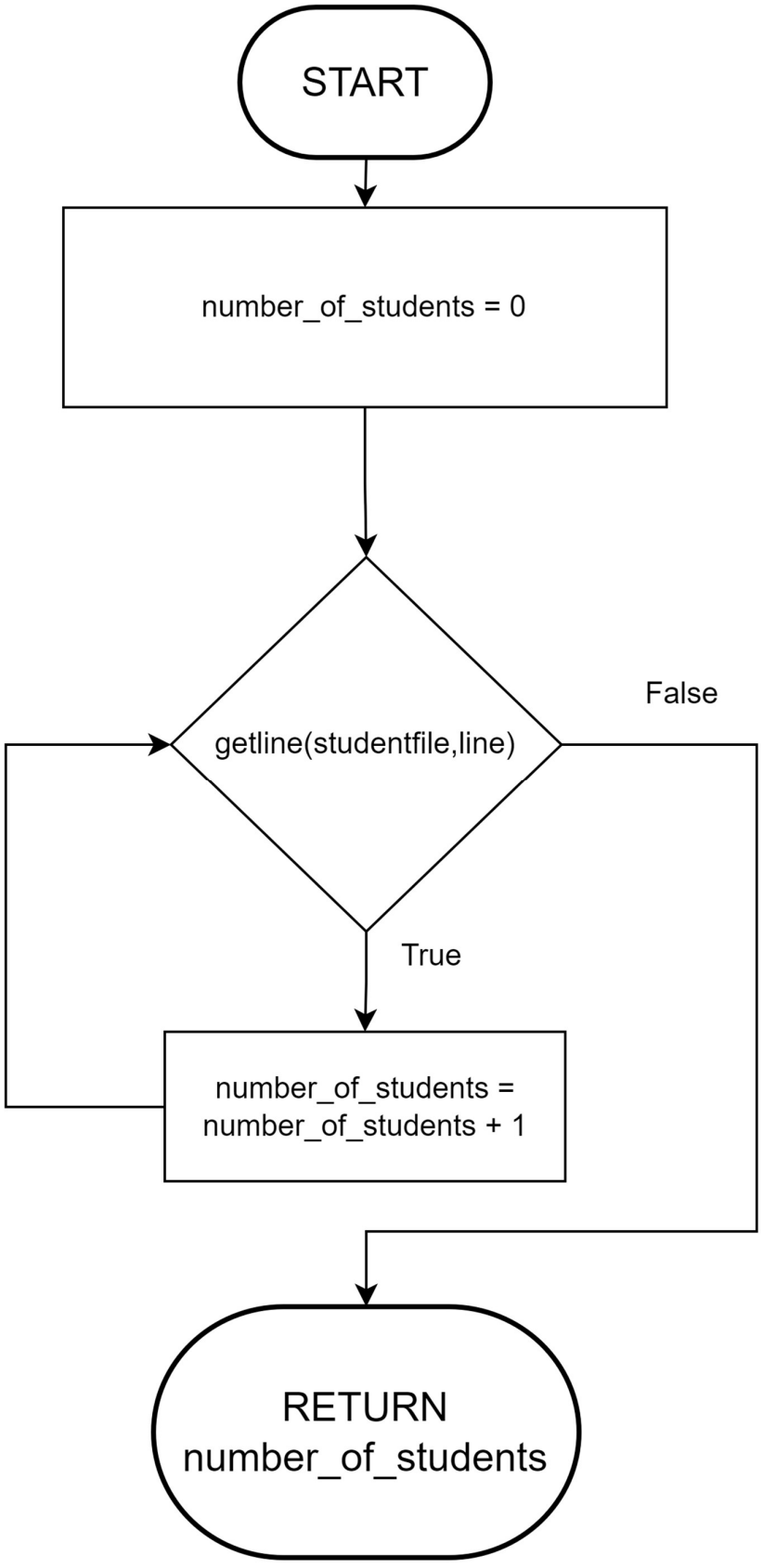


checkDuplicateStudent()

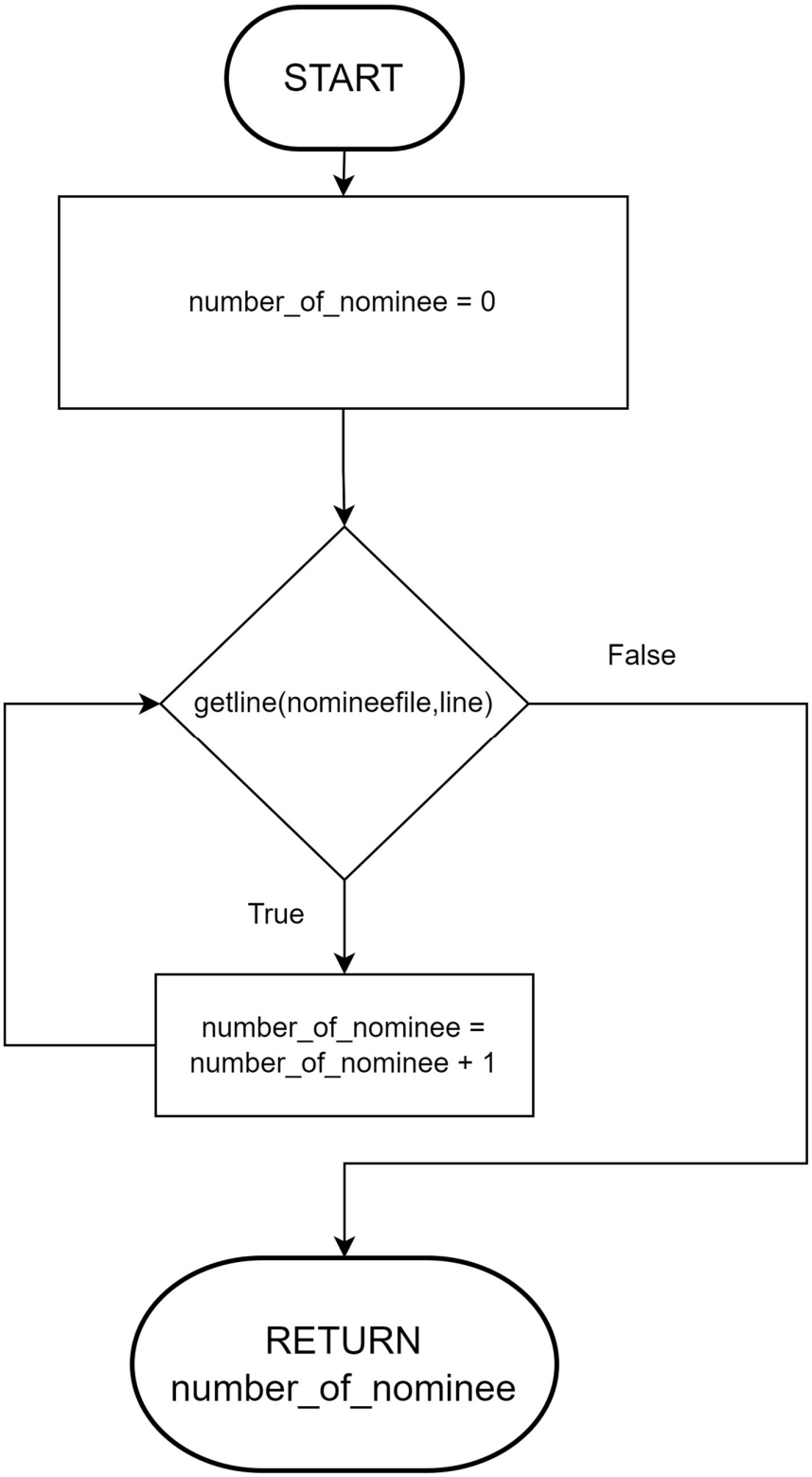




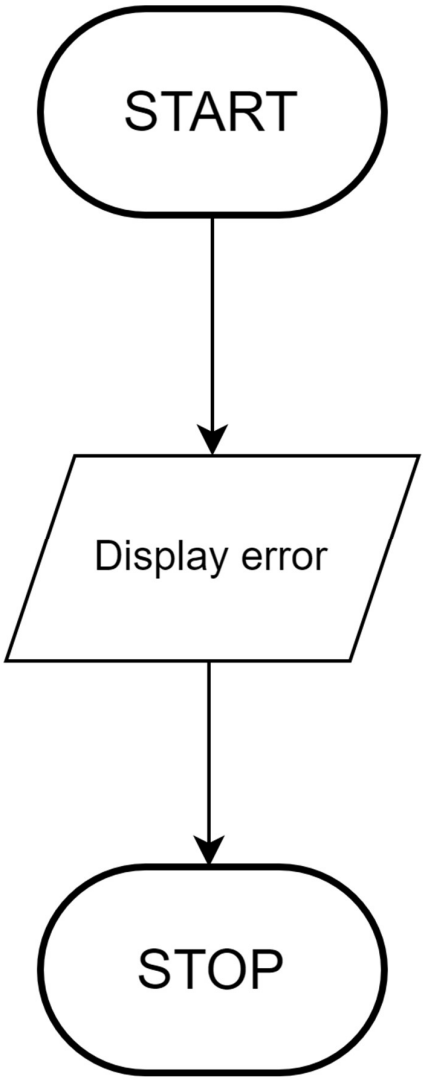
readNumStudentsFile()



readNumNomineeFile()



clearStudentFile()



Program Testing & Outputs

Module	Test Data	Expected result	Actual result reference
Admin level AdminLogin	Username: Yam Jason Password: 2222	Enter correct Admin username and password, lead to admin screen	Refer Figure 1
	Username: Bryant Password: 123	Entered wrong Admin username & password	Refer Figure 2
Admin Level AdminScreen	Option 1	Lead to Register Student Page	Refer to Figure 3
	Option 2	Lead to result page	Refer to Figure 4
	Option 3	Lead to statistics page	Refer to Figure 5
	Option 4	Lead to detailed Analysis Screen Page	Refer to Figure 6
	Option 5	Exit to main menu	Refer to Figure 7
Admin Level RegisterStudent	MyKad: 030717101080 Student ID: 22WMR13660	Successfully register a student, continue to register up until 21 students.	Refer to Figure 8
	MyKad: 03071710108P Student ID: 22WMR1366P	Will be prompted to enter Mykad and Student ID again due to wrong format	Refer to Figure 9
Student Level studentLogin	MyKad: 030717101080 Student ID: 22WMR13660	Cannot log in, information entered doesn't exist or doesn't match with database .	Refer to Figure 10
	After admin have registered the student, Mykad: 030717101080 Student ID: 22WMR13660	Will be brought to student screen.	Refer to Figure 11
Student Level studentScreen	Option 1	Will be brought to nominate screen	Refer to Figure 12
	Option 2	Will be brought to vote screen, voting session will start only when there are 5 nominees	Refer to Figure 13

	Option 3	Will go back to main menu	Refer to Figure 14
Student Level nominate	MyKad: 030717101080 Student ID: 22WMR13660 Program: RDS Year of study: 2	Successfully nominates a student	Refer to Figure 15
	(Trying to nominate twice)	A message will pop up informing not allowed to nominate the second time	Refer to Figure 16
	(Trying to nominate a student that doesn't exist) Mykad: 000000000000 Student ID: 22WMR00000 Program: RDS Year of Study: 2	A message "Invalid student!" will pop up when nominating a student that doesn't exist.	Refer to Figure 17
Student Level Vote	(Vote when there are already 5 nominees) Option 1	A farewell message will pop up	Refer to Figure 18
	(When trying to vote the second time)	A message will pop up saying student is not allowed to vote twice	Refer to Figure 19

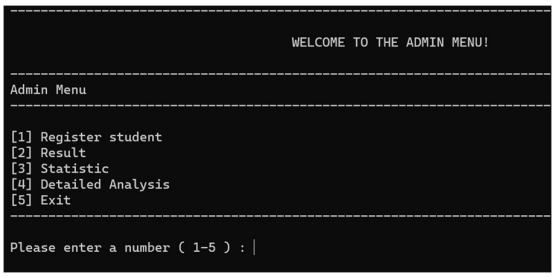


Figure 1: Successfully log in as Admin

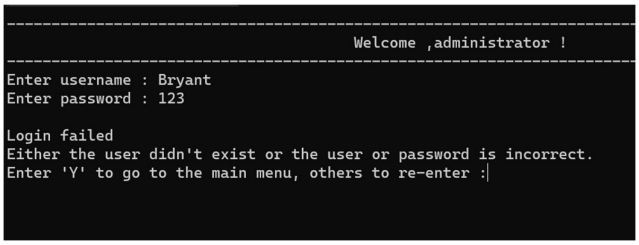


Figure 2: Failed to login as Admin

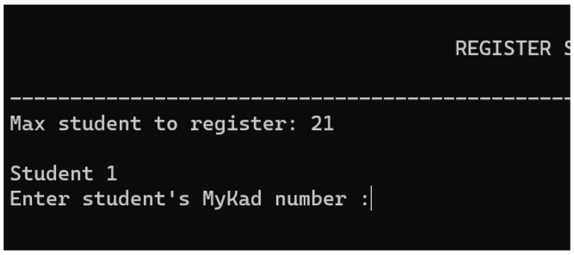


Figure 3: Screen to register students

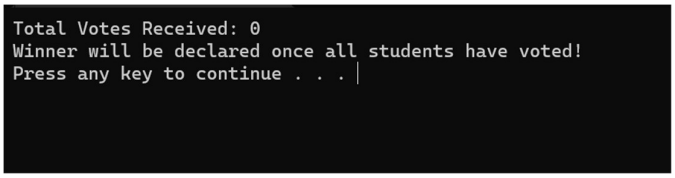


Figure 4: Result screen, winner will be declared once all students have voted.

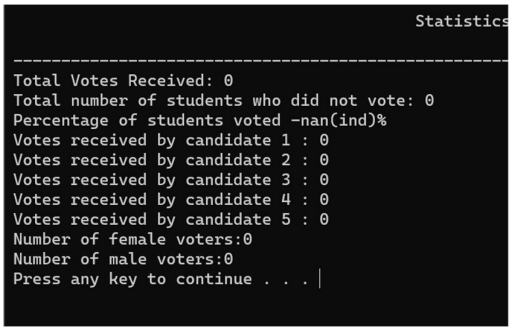


Figure 5: Statistics screen, percentage of students voted is showing error because no one has voted yet. Once, students have voted, statistics screen will be updated.

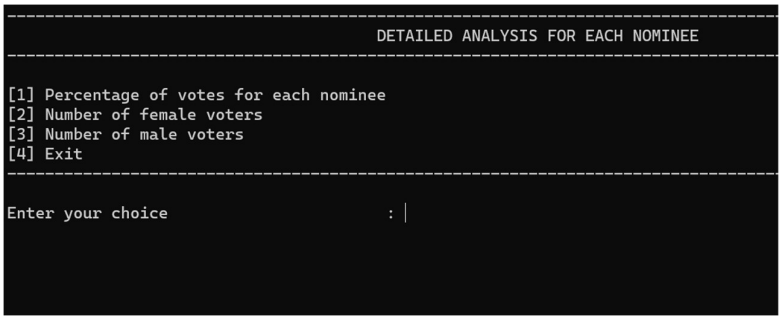


Figure 6: Detailed analysis for each nominee screen.

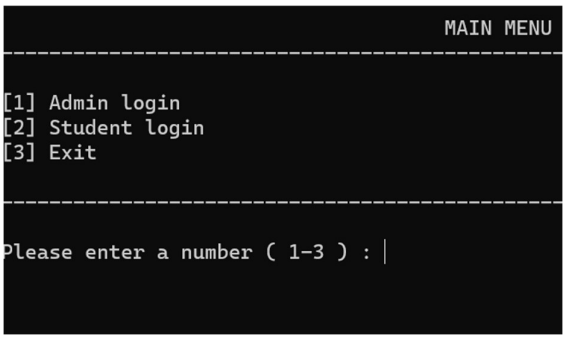


Figure 7: back to main menu.

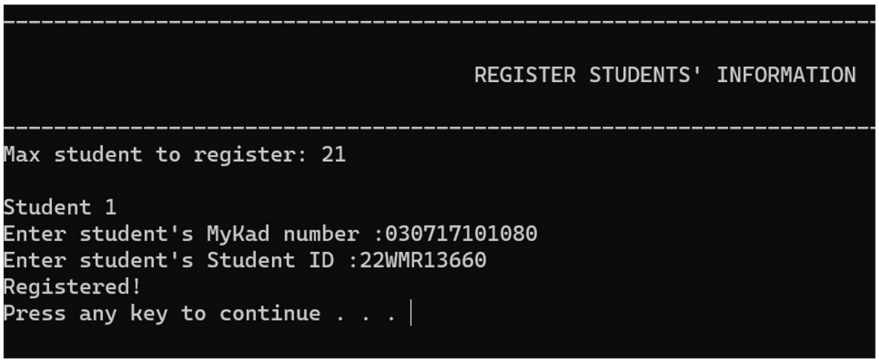


Figure 8: Registering a student

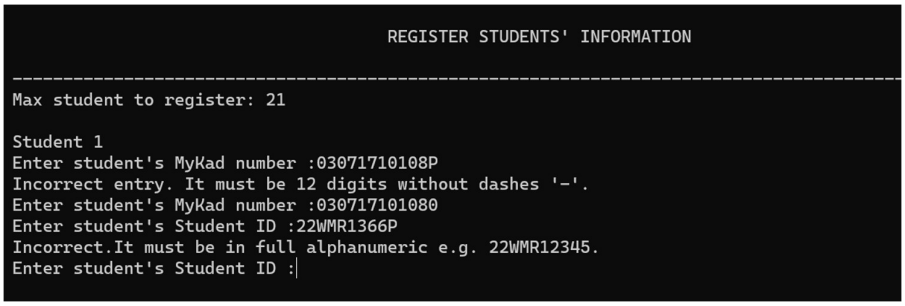


Figure 9: When admin keys in wrong format of input.

Figure 10: Login failed

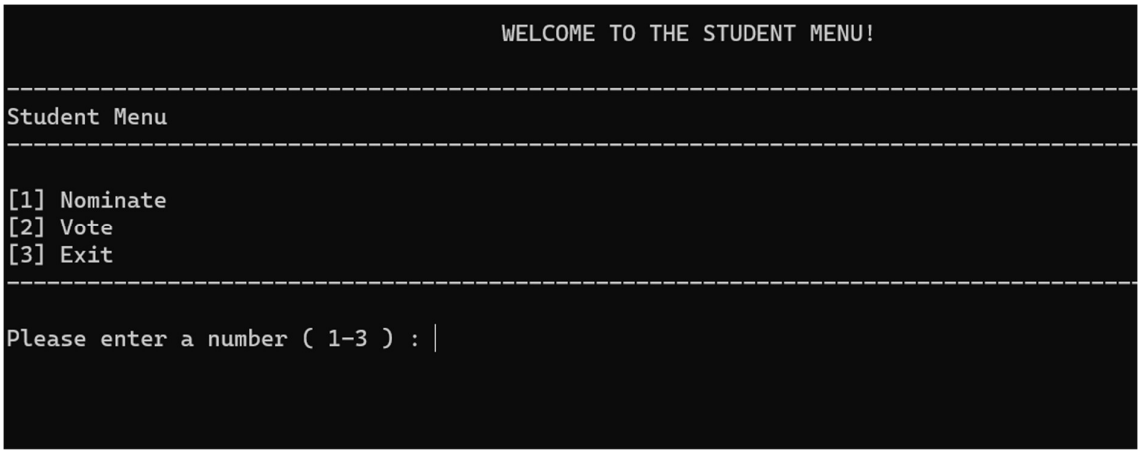


Figure 11: Only will pop up during successful login

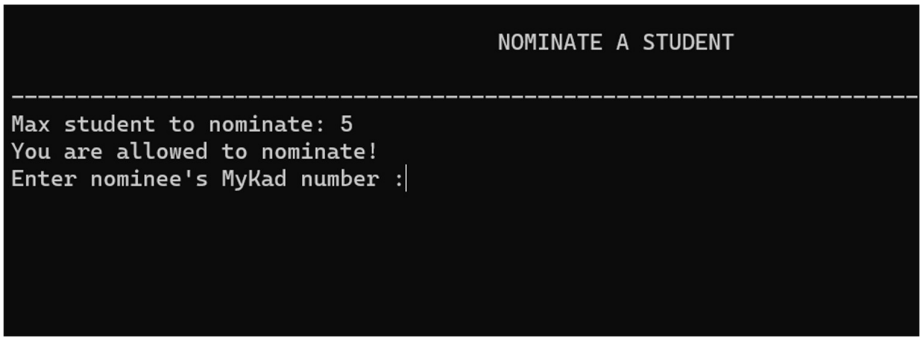


Figure 12: Nominate screen

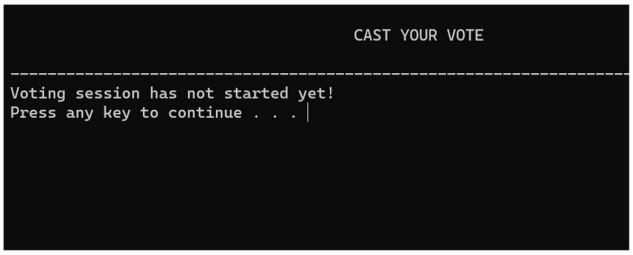


Figure 13: Vote screen

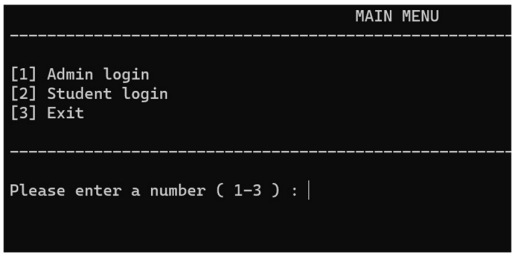


Figure 14: taken back to main menu

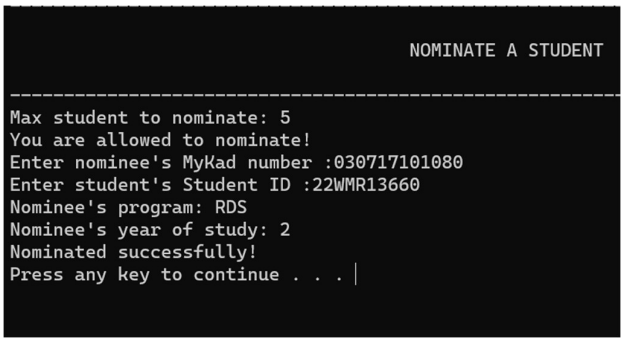


Figure 15: Nominating a student

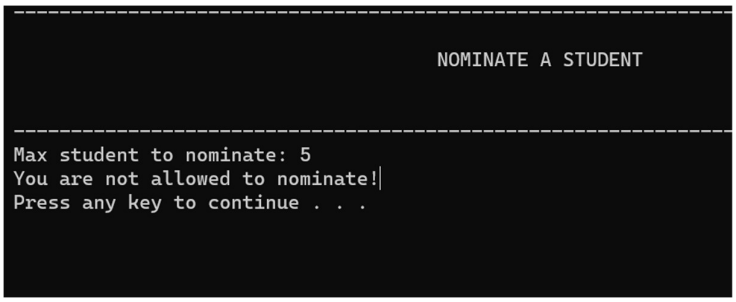


Figure 16: Not allowed to nominate twice

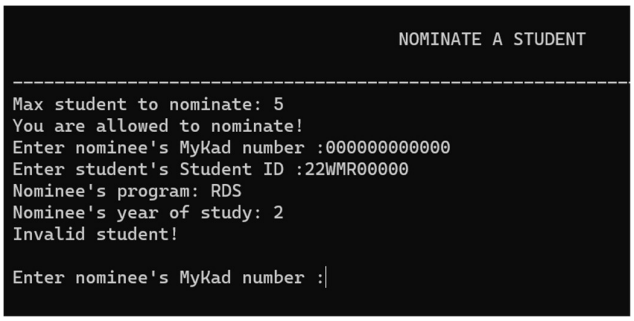


Figure 17: Cannot nominate is student that does not exist

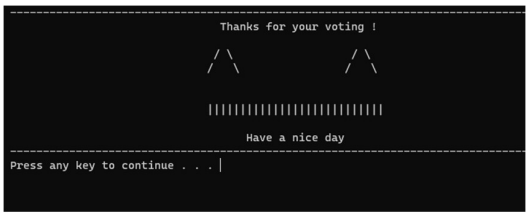


Figure 18: Farewell screen

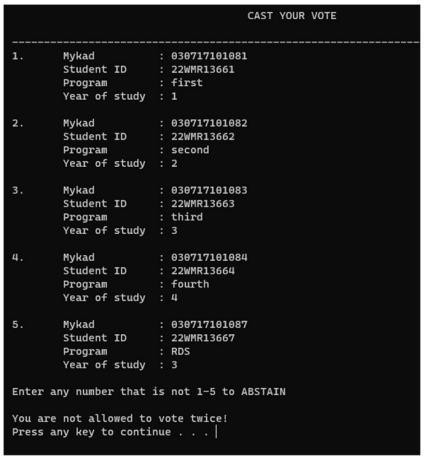


Figure 19: Attempting to vote twice

Constants & variables

Constants

Constants	Data type	Value	Purpose
numberOfAdmins	null	3	To fix the number of administrators 'information that are created and stored initially
MaxStudents	null	21	To fix the maximum number of students that can be registered by administrator in each running of program
MaxNominees	null	5	To fix the maximum number of nominees that can be nominated to close the nomination

Variables

1.Structure StudentType

Variables	Data Type	Purpose
MyKad	char	To store the value of students' MyKad into database and to be used to validate the format of MyKad according to the index of MyKad
StudentID	char	To store the value of students' Student ID into database and to be used to validate the format of Student ID according to the index of Student ID
nominateChance	bool	To set the chance of every student to nominate to true once they are registered successfully and to false once they have nominated successfully
voteChance	bool	To set the chance of every student to vote to true once they are registered successfully and to false once they have voted successfully

2.Structure NomineeType

Variables	Data Type	Purpose
n_MyKad	char	To store the value of nominees' MyKad into database and to be used to validate the format of MyKad according to the index of MyKad
n_StudentID	char	To store the value of nominees' Student ID into database and to be used to validate the format of Student ID according to the index of Student ID
program	string	To store the value of nominees' program
yearOfStudy	Int	To store the value of nominees' year of study
vote	Int	To store the value of votes received by each nominee which is initialized to 0 at first and will be incremented when nominee receives vote
femalevote	Int	To store the value of female votes received by each nominee which is initialized to 0 at first and will be incremented when nominee receives vote from female
malevote	int	To store the value of male votes received by each nominee which is initialized to 0 at first and will be incremented when nominee receives vote from male

3.main()

Variables	Data Type	Purpose
choice	int	To store the choice of user to switch to either student level ,administrator level or exit screen.
number_of_students	int	To store the number of students that read from file
number_of_nominee	int	To store the number of nominees that read from file
currentStudent	int	To store the index(position) of currently logged in student in database

4.AdminLogin()

Variables	Data Type	Purpose
isValidAdmin	bool	To check whether the administrator who is trying to login is a valid administrator by setting it to false if the administrator enters incorrect username /password and vice versa
gototMainMenu	char	To store the input value from administrator which could lead them to main menu / to continue logging in
reKeyIn	bool	To check whether the administrator have to re-key in their username and password again
AdminUsername	string	To store the input value of administrator's username and can be used to compare with the data in array later.
AdminPassword	string	To store the value of administrator's password and can be used to compare with the data in array later.
usernameArray	string	To initialize and store the administrators' username in array which served as a database
passwordArray	string	To initialize and store the administrators' password in array which served as a database

5.studentScreen()

Variables	Data Type	Purpose
studentChoice	int	To store the input value of student's choice of either going to nominate, vote or exit

6.adminScreen()

Variables	Data Type	Purpose
adminChoice	int	To store the input value of administrator's choice of either going to register student, result, statistics, detailed analysis or exit page.

7.RegisterStudent()

Variables	Data Type	Purpose
newNumStudents	int	To store the number of students that read from file and to be passed as a parameter to checkDuplicateStudent() function
MyKadformat	bool	To check the format of MyKad entered by administrator during registration
IDformat	bool	To check the format of Student ID entered by administrator during registration
studentCheck	bool	To check whether the information entered by administrator is relevant or not and let the user to re-enter if it is false. Eg: It will be set to false when there is duplicate information and vice versa.

8.studentLogin()

Variables	Data Type	Purpose
MyKadformat	bool	To check the format of MyKad entered by student during logging in
IDformat	bool	To check the format of Student ID entered by student during logging in
isValidStudent	bool	To check whether the student who is trying to login is a valid student by setting it to false if the student enters incorrect MyKad /Student ID and vice versa
reKeyIn	bool	To check whether the student have to re-key in their MyKad and Student ID again
MyKad	char	To store the input value of students' MyKad and to be used to validate the format of MyKad according to the index of MyKad
StudentID	char	To store the input value of students' Student ID and to be used to validate the format of Student ID according to the index of Student ID
gotoMainMenu	char	To store the input value from student which could lead them to main menu / to continue logging in

9.nominate()

Variables	Data Type	Purpose
i	int	To store the value of parameter -number of nominee that is passed from main function
MyKadformat	bool	To check the format of MyKad entered by student during nomination
IDformat	bool	To check the format of Student ID entered by student during nomination
yearOfStudyformat	bool	To check the format of year of study entered by student during nomination
nomineeCheck	Bool	To check whether the information entered by student is relevant or not and let the student to re-enter if it is false. Eg: It will be set to false when there is duplicate information or the nominee is not a student and vice versa.

10.checkuser()

Variables	Data Type	Purpose
nomineeIsUser	bool	To check whether is the nominee is one of the registered students and will be set to true if she/he is a registered student.

11.checkDuplicate()

Variables	Data Type	Purpose
duplicate	bool	To check whether is the nominee has been nominated before and will be set to true it there is a duplication.

12.vote()

Variables	Data Type	Purpose
choice	int	To store the input value of student's choice to vote to which nominee
lastDigitMyKad	char	To store the value of the logged in student's 12nd character in MyKad
convertedlastDigitMyKad	int	To convert the lastDigitMyKad from char data type to int data type

13.result()

Variables	Data Type	Purpose
total_votes	int	To store the value of total votes received by every nominees and is initialized to zero
topNominee	int	To store the index value of the nominee (position in file) that got the most votes, and will be assigned to sentinel value if there is a tie
highest_vote	int	To store the value of highest votes obtained by nominee among the other nominees.

14.statistics()

Variables	Data Type	Purpose
totalFemaleVotes	int	To store the value of total votes that is contributed by female voters, and it is initialized to zero.
totalMaleVotes	int	To store the value of total votes that is contributed by male voters, and it is initialized to zero.
totalvotes	int	To store the value of total votes that is contributed by all voters who do not abstain, and it is initialized to zero.

15.detailedAnalysisScreen()

Variables	Data Type	Purpose
choice	int	To store the option input by administrator to view which type of data they wish to
labelOfNominee	int	To store the input value from administrator to let them decide which nominee ‘s information they want to view
actualNumOfNomineeInFile	int	To store the value of the index (position)of nominee in file

16.checkDuplicarteStudent()

Variables	Data Type	Purpose
duplicate	bool	To check whether is the student has been registered before and will be set to true it there is a duplication.

17.readNumStudentsFile()

Variables	Data Type	Purpose
number_of_students	int	To store the value of number of students that is read from student file
line	string	To receive the input of the entire line

18.readNumNomineeFile()

Variables	Data Type	Purpose
number_of_nominee	int	To store the value of number of nominees that is read from nominee file
line	string	To receive the input of the entire line

Extra Features

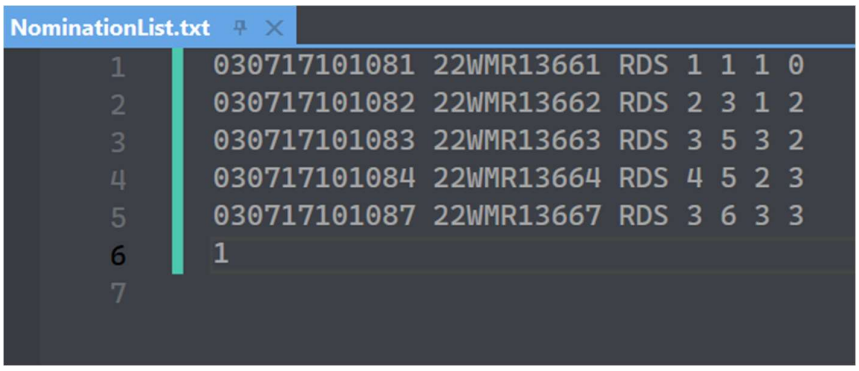
Extra Feature 1: Detail Analysis

A screen for the administrator which shows a more detailed analysis of the results where a filtering function is provided where for each candidate, the administrator can choose the option on what type of data they wish to view. For example, the admin can view the percentage of votes for each nominee, number of female and male voters of each nominee.

```
--------------------------------------------------------------------------------
                                    DETAILED ANALYSIS FOR EACH NOMINEE
--------------------------------------------------------------------------------

[1] Percentage of votes for each nominee
[2] Number of female voters
[3] Number of male voters
[4] Exit
--------------------------------------------------------------------------------

Enter your choice                    : 1
Enter the label of candidates        : 1
The percentage of votes for nominee 1   : 5.00%

Enter your choice                    : 2
Enter the label of candidates        : 1
Number of female voters for nominee 1   : 1

Enter your choice                    : 3
Enter the label of candidates        : 1
Number of male voters for nominee   1   : 0

Enter your choice                    : 1
Enter the label of candidates        : 5
The percentage of votes for nominee 5   : 30.00%

Enter your choice                    : 2
Enter the label of candidates        : 5
Number of female voters for nominee 5   : 3

Enter your choice                    : 3
Enter the label of candidates        : 5
Number of male voters for nominee   5   : 3

Enter your choice                    :
```

Extra Feature 2: Applying File as a storage medium for the system.

Allows admins to store Nominated students' details inside a file. When the program restarts, progress will not be lost. Admin will be able to change the details of nominee manually in the file if required.



Program Listing

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <iomanip>
#define numberOfAdmins 3
#define MaxStudents 21
#define MaxNominees 5
using namespace std;

struct StudentType
{
	char MyKad[13];
	char StudentID[11];
	bool nominateChance;
	bool voteChance;
};
struct NomineeType
{
	char n_MyKad[13];
	char n_StudentID[11];
	string program;
	int yearOfStudy;
	int vote;
	int femalevote;
	int malevote;
};

void mainScreen();
void AdminLogin();
void welcomeAdmin();
void RegisterStudent(StudentType student[], int* studentNum);
void displayRegisterTitle();
int adminScreen();
void adminMenu();
int studentScreen();
```

```
void studentMenu();
void studentLogin(StudentType student[], int* currentStudent);
int checkDuplicate(NomineeType nominee[], int nomineeNo);
void vote(int number_of_nominee, NomineeType nominee[], StudentType student[], int* currentStudent);
void nominate(NomineeType nominee[], int* nomineeNum, StudentType student[], int* currentStudent);
int checkuser(StudentType student[], NomineeType nominee[], int nomineeNo);
void displayNominee(NomineeType nominee[]);
void winnerDeclaration(NomineeType nominee[], int topNominee);
void result(NomineeType nominee[]);
void statistics(NomineeType nominee[], StudentType student[]);
void farewellStudents();
void detailedAnalysisScreen(NomineeType nominee[]);
void readNominationFile(NomineeType nominee[]);
void readStudentFile(StudentType student[]);
void updateStudentFile(StudentType student[]);
void updateNominationFile(NomineeType nominee[]);
int checkDuplicateStudent(StudentType student[], int studentNo);
int readNumStudentsFile();
int readNumNomineeFile();
void clearStudentFile();



int main()
{
	int choice, number_of_students, number_of_nominee, currentStudent;
	StudentType student[MaxStudents];
	NomineeType nominee[MaxNominees];


	do {
		system("cls");
		number_of_students = readNumStudentsFile();
		number_of_nominee = readNumNomineeFile();

		cout << "Number of registered student: " << number_of_students << endl;
```

```
cout << "Number of registered Nominee: " << number_of_nominee << endl;

//cout << "Number of registered nominee: " << number_of_nominee << endl;

mainScreen();
cout << "Please enter a number ( 1-3 ) : ";
cin >> choice;
system("cls");
switch (choice)
{
case 1:
        //Admin section
         //validate
        AdminLogin();
        switch (adminScreen())
        {
        case 1:
        {
                RegisterStudent(student, &number_of_students);
                break;
        }
        case 2:
                result(nominee);
                break;
        case 3:
                statistics(nominee, student);
                break;
        case 4:
                detailedAnalysisScreen(nominee);
                break;
        case 5:
                break;

        }
        break;

case 2:
```

```
                {

                        studentLogin(student, &currentStudent);
                        switch (studentScreen())
                        {
                        case 1:
                                nominate(nominee, &number_of_nominee, student, &currentStudent);
                                break;
                        case 2:
                                vote(number_of_nominee, nominee, student, &currentStudent);
                                system("pause");
                                break;
                        case 3:
                                break;
                        }

                        break;

                }
                case 3:
                {
                        cout << "You are exiting....." << endl;
                        break;
                }

                }

        } while (choice != 3);
        return 0;
}

//Main screen
void mainScreen()
{

        cout << "--------------------------------------------------------------------------------------------" <<
endl;
        cout << "                              MAIN MENU                                     " << endl;
```

```
        cout << "-----------------------------------------------------------------------------------------------\n\n";
         cout << "[1] Admin login" << endl;
         cout << "[2] Student login" << endl;
         cout << "[3] Exit" << endl << endl;
        cout << "-----------------------------------------------------------------------------------------------\n\n";


}

//AdminLogin
void AdminLogin()
{
        bool isValidAdmin;
        char gotoMainMenu;
        bool reKeyIn;
        string AdminUsername, AdminPassword;
        string usernameArray[3] = { "Wong Yee En","Yam Jason","Tang Wen Zhi" }; // Array database for usernames of admins
        string passwordArray[3] = { "1111","2222","3333" };// Array database for passwords of admins

        //Enter the loop at least once
        do
        {
                system("cls");
                cin.ignore();
                welcomeAdmin(); //Display Welcome Screen to Admin
                cout << "Enter username : "; //Prompt and read Admin Username
                getline(cin, AdminUsername);

                cout << "Enter password : ";//Prompt and read Admin Password
                getline(cin, AdminPassword);

                // Comparing the input with array database
                for (int i = 0; i < numberOfAdmins; i++)
                {
                        //If the input matches with database
                        if (AdminUsername == usernameArray[i] && AdminPassword == passwordArray[i])
                        {
```

```
                        //She or He is one of the admins
                        isValidAdmin = true;
                        break;
                }

                else
                        //She or He is not one of the admins
                        isValidAdmin = false;

        }
        //When She or He is not one of the admins
        if (isValidAdmin == false)
        {
                //She or He is not able to log in
                cout << "\nLogin failed" << endl;
                cout << "Either the user didn't exist or the user or password is incorrect." <<
endl;
                //Prompt and get whether She or He wants to go back to main menu / re-enter
                cout << "Enter 'Y' to go to the main menu, others to re-enter :";
                cin >> gotoMainMenu;
                //If input is 'Y'/ 'y'
                if (gotoMainMenu == 'Y' || gotoMainMenu == 'y')
                {
                        system("cls");
                        main();// Go to main function
                }
                //If the input is not 'Y' / 'y'
                else {
                        reKeyIn = true;  // Set to let the admin re-enter later
                }

        }
        //When She or He is admin
        else
        {
                // Set admin don't have to re-enter later
                reKeyIn = false;
                break;
```

```
                }

                //If re-enter is true, loop again
        } while (reKeyIn == true);

}

//Student Screen
int studentScreen() {
        int studentChoice;
        system("cls");

        //Display Student Menu
        studentMenu();

        //Prompt and get choice
        cout << "Please enter a number ( 1-3 ) : ";
        cin >> studentChoice;

        //Prompt and get choice again if the input is not within the range of 1 - 3
        while (studentChoice > 3 || studentChoice < 1)
        {
                cout << "Invalid choice!" << endl;
                cout << "Enter again: ";
                cin >> studentChoice;
        }
        system("cls");
        return studentChoice;
}

//Student Menu
void studentMenu()
{
        cout << "-----------------------------------------------------------------------------------------------" <<
endl << endl;
        cout << "                        WELCOME TO THE STUDENT MENU!                        "
<< endl << endl;
```

```
	cout << "-----------------------------------------------------------------------------------------------" <<
endl;
	cout << "Student Menu" << endl;
	cout << "-----------------------------------------------------------------------------------------------\n\n";
	cout << "[1] Nominate" << endl;
	cout << "[2] Vote" << endl;
	cout << "[3] Exit" << endl;
	cout << "-----------------------------------------------------------------------------------------------\n\n";
}

int adminScreen() {
	int adminChoice;
	system("cls");

	//Display Admin Menu
	adminMenu();

	//Prompt and get choice
	cout << "Please enter a number ( 1-5 ) : ";
	cin >> adminChoice;

	//Prompt and get choice again if the input is not within the range of 1 - 3
	while (adminChoice > 5 || adminChoice < 1)
	{
		cout << "Invalid choice!" << endl;
		cout << "Enter again: ";
		cin >> adminChoice;
	}
	system("cls");
	return adminChoice;
}

void adminMenu()
{
	cout << "-----------------------------------------------------------------------------------------------" <<
endl << endl;
	cout << "                              WELCOME TO THE ADMIN MENU!                              "
<< endl << endl;
```

```
	cout << "-------------------------------------------------------------------------------------------" <<
endl;
	cout << "Admin Menu" << endl;
	cout << "-------------------------------------------------------------------------------------------\n\n";
	cout << "[1] Register student" << endl;
	cout << "[2] Result" << endl;
	cout << "[3] Statistic" << endl;
	cout << "[4] Detailed Analysis" << endl;
	cout << "[5] Exit" << endl;
	cout << "-------------------------------------------------------------------------------------------\n\n";
}

void RegisterStudent(StudentType student[], int* studentNum)
{
	int i, newNumStudents;
	bool MyKadformat = false; // Initialize MyKadformat to false
	bool IDformat = false; // Initialize MyKadformat to false
	bool studentCheck = false;// Initialize studentCheck to false

	//Display Register Title
	displayRegisterTitle();
	//If the students
	if (MaxStudents - *studentNum == 0)
		system("pause");
	// If the number of students in file is not equal to maximum students to be registered.

	//
	clearStudentFile();

	cin.ignore(100, '\n');

	//Loop
	for (i = 0; i < MaxStudents; i++)
	{
		cout << "\nStudent " << i + 1 << endl;
		do
		{
			//MyKad Input and Validation
```

```
//Enter this loop at least once
do
{

        //Prompt and get student's MyKad number
        cout << "Enter student's MyKad number :";
        cin.getline(student[i].MyKad, 13);

        //MyKad Validation
        //It must be 12 characters
        if (strlen(student[i].MyKad) != 12)
        {
                //Set MyKadformat to false if there are no 12 characters
                MyKadformat = false;
        }

        else
        {
                //Set MyKadformat to true if there are 12 characters
                MyKadformat = true;
                //Use loop to check each character in MyKad Number
                for (int j = 0; j < 12; j++)
                {
                        //If one of the characters in MyKad Number is not a
digit
                        if (isdigit(student[i].MyKad[j]) == 0)
                        {
                                //Set MyKadformat to false
                                MyKadformat = false;
                                break;
                        }

                }
        }
        //If MyKadformat is false
        if (MyKadformat == false)
                //Display the correct format
```

```
                                        cout << "Incorrect entry. It must be 12 digits without dashes '-
'." << endl;

                                //Loop again while MyKadformat is false.
                        } while (MyKadformat == false);

                        //Student ID Input and Validation
                        //Enter this loop at least once
                        do
                        {
                                //Prompt and get student's Student ID
                                cout << "Enter student's Student ID :";
                                cin.getline(student[i].StudentID, 11);

                                //Student ID Validation
                                //It must be 10 characters
                                if (strlen(student[i].StudentID) != 10)
                                        //Set IDformat to false if there are no 10 characters
                                        IDformat = false;
                                else
                                {
                                        //Set IDformat to true if there are 10 characters
                                        IDformat = true;
                                        if      (isdigit(student[i].StudentID[0])       ==      0       ||
isdigit(student[i].StudentID[1]) == 0)
                                                //Set IDformat to false if the first and second character
are not digits
                                                IDformat = false;
                                        else
                                        {
                                                //Set IDformat to true if the first and second character
are digits
                                                IDformat = true;
                                                //Loop from the third character to the fifth character
of Student ID
                                                //to check whether they are alphabets in uppercase
                                                for (int k = 2; k < 5; k++)
                                                {
```

```
                                        if (isalpha(student[i].StudentID[k]) == 0 ||
isupper(student[i].StudentID[k]) == 0)
                                        {
                                                //If the characters are not alphabets or
in uppercase
                                                //Set IDformat to false
                                                IDformat = false;
                                                break;
                                        }
                                        else
                                        {
                                                //If the characters are alphabets and in
uppercase
                                                //Set IDformat to true
                                                IDformat = true;
                                                //Check the 6th to 10th characters
using loop
                                                for (int s = 5; s < 10; s++)
                                                {
                                                        //If any of the charaters is not
a digit
                                                        if
(isdigit(student[i].StudentID[s]) == 0)
                                                        {
                                                                //Set IDformat to
false
                                                                IDformat = false;
                                                                break;
                                                        }
                                                }
                                        }
                                }

                        }

                }
                //Display correct format if IDformat is false.
                if (IDformat == false)
                        cout << "Incorrect.It must be in full alphanumeric e.g.
22WMR12345." << endl;
```

```
                //Loop again when IDformat is false
        } while (IDformat == false);


        //Read number of students in the file
        newNumStudents = readNumStudentsFile();

        // If the input is duplicate with the any one of the record in database
        if (checkDuplicateStudent(student, newNumStudents) == 1)
        {
                //Set studentCheck to false
                studentCheck = false;
        }
        else
        {
                //Set studentCheck to true if there is no duplication
                studentCheck = true;
        }

        //Loop again when the studentCheck is false
} while (studentCheck == false);



student[i].nominateChance = true; //Set student's nominate chance to true in array
student[i].voteChance = true;//Set student's vote chance to true in array
cout << "Registered!" << endl;
system("pause");

//Append RegisterStudent text file
ofstream outData;
outData.open("RegisterStudent.txt", ios_base::app | ios_base::out);
//The file cant be opened
if (!outData)
{
        //Display error message
        cout << "Error opening file";
```

```
                }
                else
                {
                        //Write from array to file
                        outData << student[i].MyKad << " " << student[i].StudentID << " " << 
student[i].nominateChance << " " << student[i].voteChance << endl;

                }
                //Close file
                outData.close();



        }
        system("cls");
}

void displayRegisterTitle()
{
        cout << "-------------------------------------------------------------------------------------------" << 
endl << endl;
        cout << "                        REGISTER STUDENTS' INFORMATION                        " 
<< endl << endl;
        cout << "-------------------------------------------------------------------------------------------" << 
endl;
        cout << "Max student to register: " << MaxStudents << endl;
}

void studentLogin(StudentType student[], int* currentStudent)
{
        bool MyKadformat = false, IDformat = false, isValidStudent = false, reKeyIn = false;
        char MyKad[13], StudentID[11], gotoMainMenu;



        do
        {
                cout << "-------------------------------------------------------------------------------------------
----" << endl;
```

```
		cout << "                    WELCOME STUDENT !                    "
<< endl;
		cout << "--------------------------------------------------------------------------------------------
----" << endl;

		//Input MyKad at least once
		do
		{
			cin.ignore(100, '\n');
			//Prompt and get MyKad number
			cout << "Enter MyKad number:";
			cin.getline(MyKad, 13);

			//MyKad Validation
			//If there are no 12 characters in MyKad
			if (strlen(MyKad) != 12)
			{
				//Set MyKadformat to false
				MyKadformat = false;
			}

			else
			{
				//Set MyKadformat to true if there are 12 characters in MyKad
				MyKadformat = true;
				//Loop to check every character
				for (int j = 0; j < 12; j++)
				{
					//If one of the character is not digit
					if (isdigit(MyKad[j]) == 0)
					{
						//Set MyKadformat to false
						MyKadformat = false;
						break;
					}

				}
			}
```

```
                //If Mykadformat is false
                if (MyKadformat == false)
                        //Display error message with correct format of Mykad number
                        cout << "Incorrect entry. It must be 12 digits without dashes '-'." <<
endl;
                //Loop again when MyKadformat is false
        } while (MyKadformat == false);

        //Student ID Input and Validation
        do
        {
                //Prompt and get Student ID
                cout << "Enter student's Student ID :";
                cin.getline(StudentID, 11);

                //Student ID Validation
                //If there are no 10 characters in Student ID
                if (strlen(StudentID) != 10)
                        //Set IDformat to false
                        IDformat = false;
                else
                {
                        //If there are 10 characters
                        //Set IDfformat to true
                        IDformat = true;
                        //If the one of the 1st and 2nd characters is not digit
                        if (isdigit(StudentID[0]) == 0 || isdigit(StudentID[1]) == 0)
                                //Set IDformat to false
                                IDformat = false;
                        else
                        {
                                //Set IDformat to true if the both 1st and 2nd characters are
digits
                                IDformat = true;
                                //Loop to check the 3rd to 5th characters
                                for (int k = 2; k < 5; k++)
                                {
```

```
                                        //If one of the character is not alphabet or is not in uppercase`
                                        if (isalpha(StudentID[k]) == 0 || isupper(StudentID[k]) == 0)
                                        {
                                                //Set IDformat to false
                                                IDformat = false;
                                                break;
                                        }
                                        else
                                        {
                                                //Set IDformat to true if 3rd to 5th characters are alphabets in uppercase
                                                IDformat = true;
                                                //Loop from the 6th to 10th characters
                                                for (int s = 5; s < 10; s++)
                                                {
                                                        //If they are not digits
                                                        if (isdigit(StudentID[s]) == 0)
                                                        {
                                                                //Set IDformat to false
                                                                IDformat = false;
                                                                break;
                                                        }
                                                }
                                        }
                                }

                        }

                }
                //If IDformat is false
                if (IDformat == false)
                        //Display error message and show the correct format of Student ID
                        cout << "Incorrect.It must be in full alphanumeric e.g. 22WMR12345." << endl;
                //Loop again when IDformat is false
        } while (IDformat == false);
```

```
		//compare with database
		//Read students' file
		ifstream inData;
		inData.open("RegisterStudent.txt");
		//If cant open the text file
		if (!inData)
		{
			//Display error message
			cout << "Error opening data file" << endl;
		}
		else
		{
			//Enter loop if open the file succesfully
			//Loop to check whether the input matches with data in the database
			for (int i = 0; i < MaxStudents; i++)
			{
				inData >> student[i].MyKad >> student[i].StudentID >> student[i].nominateChance >> student[i].voteChance;
				//If any of the record in database matches with the input 's MyKad and StudentID
				if (strcmp(MyKad, student[i].MyKad) == 0 && strcmp(StudentID, student[i].StudentID) == 0)
				{
					isValidStudent = true;//Set isValidStudent to true
					*currentStudent = i; // Let the matched student's position in file into pointer variable *currentStudent
					break;
				}
				else
				{
					//Set isValidStudent to false if there is not matching
					isValidStudent = false;
				}

			}
		}
		//close file
```

```
                inData.close();

                //if does not match with database
                if (isValidStudent == false)
                {
                        //Display login failed message
                        cout << "\nLogin failed" << endl;
                        cout << "Either the user didn't exist or the user or password is incorrect." <<
endl;
                        cout << "Enter 'Y' to go to the main menu, others to re-enter :"; //Prompt and
get gotoMainMenu
                        cin >> gotoMainMenu;
                        //If input is 'Y' / 'y'
                        if (gotoMainMenu == 'Y' || gotoMainMenu == 'y')
                        {
                                system("cls");
                                //Go to main function
                                main();
                        }
                        else {
                                //If input is other than 'Y' & 'y'
                                //Set reKeyIn to true
                                reKeyIn = true;
                        }

                }

                else
                {
                        //If match with database
                        //Set reKeyIn to false
                        reKeyIn = false;
                        break;
                }
                //Loop again when reKeyIn is true
        } while (reKeyIn == true);

}
```

```
//Nominate function
void nominate(NomineeType nominee[], int* nomineeNum, StudentType student[], int* currentStudent)
{
	//Initialize i to number of registered nominee
	int i = *nomineeNum;
	bool MyKadformat = false;
	bool IDformat = false;
	bool yearOfStudyformat = false;
	bool nomineeCheck = false;


	cout << "-----------------------------------------------------------------------------------------------" <<
endl << endl;
	cout << "                              NOMINATE A STUDENT                                " << endl
<< endl;
	cout << "-----------------------------------------------------------------------------------------------" <<
endl;
	cout << "Max student to nominate: " << MaxNominees << endl;
	//If number of registered nominee has reached the maximum number of nominees (include
abstain record in file)
	if (i == MaxNominees + 1)
	{
		//Display warning message
		cout << "Max nominee has been reached!" << endl;


	}
	//If the logged in student 's nominate chance is true (havent nominate)
	else if (student[*currentStudent].nominateChance == 1)
	{

		cout << "You are allowed to nominate!" << endl;


		//MyKad Input and Validation
		do
		{

			cin.ignore();
			do
			{
```

```
                    //MyKad Input
                    cout << "Enter nominee's MyKad number :";
                    cin.getline(nominee[i].n_MyKad, 13);

                    //MyKad Validation
                    //If there are no 12 characters in MyKad
                    if (strlen(nominee[i].n_MyKad) != 12)
                    {
                        //Set MyKadformat to false
                        MyKadformat = false;
                    }

                    else
                    {
                        //Set MyKadformat to true if there are 12 characters in MyKad
                        MyKadformat = true;
                        //Loop to check every character in MyKad
                        for (int j = 0; j < 12; j++)
                        {
                            //If one of the character in MyKad is not a digit
                            if (isdigit(nominee[i].n_MyKad[j]) == 0)
                            {
                                //Set MyKadformat to false
                                MyKadformat = false;
                                break;
                            }

                        }
                    }
                    //If MyKadformat is false
                    if (MyKadformat == false)
                        //Display warning message
                        cout << "Incorrect entry. It must be 12 digits without dashes '-
'." << endl;

                    //Loop again while MyKadformat is false
                } while (MyKadformat == false);
```

```
                    //Student ID Input and Validation
                    do
                    {
                            //Student ID Input
                            cout << "Enter student's Student ID :";
                            cin.getline(nominee[i].n_StudentID, 11);

                            //Student ID Validation
                            //If there are no 10 characters in Student ID
                            if (strlen(nominee[i].n_StudentID) != 10)
                                    //Set IDformat to false
                                    IDformat = false;
                            else
                            {
                                    //Set IDformat to true if there are 10 characters
                                    IDformat = true;
                                    //If one of the 1st and 2nd characters is not a digit
                                    if (isdigit(nominee[i].n_StudentID[0]) == 0 ||
isdigit(nominee[i].n_StudentID[1]) == 0)
                                            //Set IDformat to false
                                            IDformat = false;
                                    else
                                    {
                                            //Set IDformat to true if both of the 1st and 2nd
characters are digits
                                            IDformat = true;
                                            //Loop to check from the 3rd to 5th characters
                                            for (int k = 2; k < 5; k++)
                                            {
                                                    //If one of the character is not alphabet in
uppercase
                                                    if (isalpha(nominee[i].n_StudentID[k]) == 0
|| isupper(nominee[i].n_StudentID[k]) == 0)
                                                    {
                                                            //Set IDformat to false
                                                            IDformat = false;
                                                            break;
                                                    }
                                                    else
```

```
                                                {
                                                        //Set IDformat to true if the
characters are alphabets in uppercase
                                                        IDformat = true;
                                                        //Loop from 6th to 10th characters
                                                        for (int s = 5; s < 10; s++)
                                                        {
                                                                //If one of the characters is
not a digit
                                                                if
(isdigit(nominee[i].n_StudentID[s]) == 0)
                                                                {
                                                                        //Set IDformat to
false
                                                                        IDformat = false;
                                                                        break;
                                                                }
                                                        }
                                                }
                                        }

                                }

                        }
                        //If IDformat is false
                        if (IDformat == false)
                                //Display error message and show correct format of Student
ID
                                cout << "Incorrect.It must be in full alphanumeric e.g.
22WMR12345." << endl;
                        //Loop again while IDformat false
                } while (IDformat == false);

                //Prompt and get nominee's program
                cout << "Nominee's program: ";
                getline(cin, nominee[i].program);

                //Enter this loop at least once
                do
```

```
                {
                        //Prompt and get Nominee 's year of study
                        cout << "Nominee's year of study: ";
                        cin >> nominee[i].yearOfStudy;
                        //If input is less than 1
                        if (nominee[i].yearOfStudy < 1)
                        {
                                //Set yearOfStudyformat to false
                                yearOfStudyformat = false;
                        }
                        else
                        {
                                //Set yearOfStudyformat to true if the input is not less than 1
                                yearOfStudyformat = true;
                        }
                        //Loop again if yearOfStudy is false
                } while (yearOfStudyformat == false);

                //check if the details entered is from registered student, need to be from registered student
                //if not from registered user, will loop again.
                if (checkuser(student, nominee, i) == 0)
                {
                        nomineeCheck = false;
                }
                //check if the details entered is from nomination list student, cannot be a duplicate.
                //if duplicate, loop again
                else if (checkDuplicate(nominee, i) == 1)
                {
                        nomineeCheck = false;
                }
                else
                {
                        nomineeCheck = true;
                }

        } while (nomineeCheck == false);
```

```
//Set the logged in student's nominate chance to false
student[*currentStudent].nominateChance = 0;

//Append nomination list
ofstream outData;
outData.open("NominationList.txt", ios_base::app | ios_base::out);
//If cant open the text file
if (!outData)
{
	//Display error message
	cout << "Error opening files. ";
}
else
{
	cout << "Nominated successfully!" << endl;
	//set votes received to 0
	nominee[i].vote = 0;
	nominee[i].femalevote = 0;
	nominee[i].malevote = 0;

	//make a profile for abstain when nomineeNum is 4 ( which means when there are 5 nominees in file)
	if (i == 4)
	{
		//set the vote of sixth nominee (which represents abstain) to zero
		nominee[5].vote = 0;
		//Write the data from array to file
		outData << nominee[i].n_MyKad << " " << nominee[i].n_StudentID << " " << nominee[i].program << " " << nominee[i].yearOfStudy << " " << nominee[i].vote << " " << nominee[i].femalevote << " " << nominee[i].malevote << endl;
		//including the vote of sixth nominee
		outData << nominee[5].vote << endl;
	}
	else
	{
```

```
                                        //If there are no 5 nominees in file
                                        //Write the data from array to file
                                        outData << nominee[i].n_MyKad << " " << nominee[i].n_StudentID
<< " " << nominee[i].program << " " << nominee[i].yearOfStudy << " " << nominee[i].vote << " " <<
nominee[i].femalevote << " " << nominee[i].malevote << endl;
                                }

                        }
                        outData.close();



                        //update RegisterStudent.txt nominate chance
                        updateStudentFile(student);



        }
        else
        {
                //If the logged in student's nominate chance is not available
                //Display warning message
                cout << "You are not allowed to nominate!" << endl;
        }
        system("pause");



}
//Check nominee is one of the registered student
int checkuser(StudentType student[], NomineeType nominee[], int nomineeNo)
{
        bool nomineeIsUser = false;
        //Read RegisterStudent text file
        ifstream inData;
        inData.open("RegisterStudent.txt");
        //If cant open the text file
        if (!inData) {
                //Display error message
                cout << "Error opening files.";
        }
```

```
	else {
		//Loop from the first student to the last student in file
		for (int i = 0; i < MaxStudents; i++)
		{
			//Read every students' record in file
			inData >> student[i].MyKad >> student[i].StudentID >> student[i].nominateChance >> student[i].voteChance;

		}
		inData.close();

		for (int i = 0; i < MaxStudents; i++)
		{
			//If there is any record from database matches with input
			if (strcmp(nominee[nomineeNo].n_MyKad, student[i].MyKad) == 0 || strcmp(nominee[nomineeNo].n_StudentID, student[i].StudentID) == 0)
			{
				//Set nomineeIsUser to true which means nominee is one of the registered students
				nomineeIsUser = 1;
				break;
			}
			else
			{
				//Set nomineeIsUser to false which means nominee is not one of the registered students
				nomineeIsUser = 0;
			}
		}
		//If nominee is not one of the user
		if (nomineeIsUser == 0)
		{
			cout << "Invalid student!" << endl << endl;
		}


	}

	//Return true if nominee is one of the registered students
```

```
	if (nomineeIsUser == 1)
		return 1;
	//Return false if nominee is not one of the registered students
	else
		return 0;
}
//Check the new nominee is not duplicate with the nominee in database
int checkDuplicate(NomineeType nominee[], int nomineeNo)
{
	bool duplicate = false;
	//If the number of nominees in file is more than zero
	if (nomineeNo > 0)
	{
		//Read NominationList text file
		ifstream inData;
		inData.open("NominationList.txt");

		if (!inData) {
			cout << "Error opening files.";
		}
		else {
			//Loop from the first nominee in the file up to the nominee in front of the new nominated nominee
			for (int i = 0; i < nomineeNo; i++)
			{
				//Read each of the nominee's records
				inData >> nominee[i].n_MyKad >> nominee[i].n_StudentID >> nominee[i].program >> nominee[i].yearOfStudy >> nominee[i].vote >> nominee[i].femalevote >> nominee[i].malevote;
				//If there is a match of MyKad and StudentID between input and database
				if (strcmp(nominee[nomineeNo].n_MyKad, nominee[i].n_MyKad) == 0 || strcmp(nominee[nomineeNo].n_StudentID, nominee[i].n_StudentID) == 0)
				{
					//Set duplicate to true
					duplicate = true;
					cout << "Nominee entered is a duplicate. Registration failed." << endl << endl;
					break;
```

```
                                        }
                                        else
                                        {
                                                //Set duplicate to false if there is no matching of MyKad and StudentID between input and database
                                                duplicate = false;
                                        }
                                }

                        }
                        inData.close();
                }
                else
                {
                        //Set duplicate to false if the number of nominee in file is not more than zero
                        duplicate = false;
                }


                if (duplicate == true)
                        return 1;
                else
                        return 0;
}

void vote(int number_of_nominee, NomineeType nominee[], StudentType student[], int* currentStudent)
{

                int choice;
                char lastDigitMyKad = student[*currentStudent].MyKad[11]; // last digit of the logged in student's MyKad
                int convertedlastDigitMyKad = (int)lastDigitMyKad; // change the data type from character to integer
                cout << "-----------------------------------------------------------------------------------------------" << endl << endl;
                cout << "                                CAST YOUR VOTE                                 " << endl << endl;
```

```
	cout << "-----------------------------------------------------------------------------------------------" << endl;
	//if the number of nominees in file is not equal to the (maximum number of nominees plus one record of abstain)
	if (number_of_nominee != MaxNominees + 1)
	{
		cout << "Voting session has not started yet!" << endl;
	}
	else
	{
		//read NominationList from text file
		readNominationFile(nominee);

		//read RegisterStudent from text file;
		readStudentFile(student);
		//Display the nominees' details
		displayNominee(nominee);

		//If the voting chance of logged in student is true
		if (student[*currentStudent].voteChance == 1)
		{
			//Display warning message
			cout << "Once a vote is cast, no amendment or changing of vote is allowed." << endl;
			//Prompt and get choice(vote) from user
			cout << "Cast your vote by entering a correct number (candidate number label) based on the number of candidates displayed on the screen \n";

			cout << "Enter your vote: ";
			cin >> choice;
			//any number that is not one of the candidate labels will get turned to 6 (which means the sixth record in file)
			if (choice < 1 || choice > 5)
			{
				choice = 6;
			}

			cout << "You have voted for candidate " << choice << endl;
```

```
//update vote count to struct array
choice -= 1;
//Increment the vote of nominee that got voted by 1
nominee[choice].vote += 1;
//Set the logged in student 's vote chance to false
student[*currentStudent].voteChance = 0;

//Differentiate gender of voters
//If the vote is not for abstain
if (choice != 5)
{
	//if the remainder of last digit of MyKad is zero when divided by 2
	if (convertedlastDigitMyKad % 2 == 0)
	{
		//Increment the female vote of voted nominee by 1
		nominee[choice].femalevote++;

	}
	else
	{
		//if the remainder of last digit of MyKad is not zero when divided by 2
		//Increment the male vote of voted nominee by 1
		nominee[choice].malevote++;

	}
}
else
{
	//If the vote is not for abstain
	cout << "YOU CHOSE TO ABSTAIN!" << endl;
}

//update vote chance in register student list
updateStudentFile(student);
```

```
                        //update vote count in nominationlist file
                        //update nominate result
                        updateNominationFile(nominee);

                        system("cls");
                        //Display farewell screen
                        farewellStudents();


                }
                else
                {
                        //If the voting chance of logged in student is false
                        cout << "You are not allowed to vote twice!" << endl;

                }


        }
}

//Display nominees for students to know the nominee's details & labels
void displayNominee(NomineeType nominee[])
{
        //Loop all the nominees in database
        for (int i = 0; i < 5; i++)
        {
                cout << i + 1 << ". \t" << setw(15) << left << "Mykad" << ": " << nominee[i].n_MyKad
<< endl;
                cout << "  \t" << setw(15) << left << "Student ID" << ": " << nominee[i].n_StudentID
<< endl;
                cout << "  \t" << setw(15) << left << "Program" << ": " << nominee[i].program <<
endl;
                cout << "  \t" << setw(15) << left << "Year of study" << ": " << nominee[i].yearOfStudy
<< endl << endl;
        }
        cout << "Enter any number that is not 1-5 to ABSTAIN" << endl << endl;
}
```

```
//Voting result
void result(NomineeType nominee[])
{
	int total_votes = 0, topNominee, highest_vote;

	//read result from Nomination file
	readNominationFile(nominee);

	//Calculate total votes
	for (int i = 0; i < 6; i++)
	{
		total_votes += nominee[i].vote;
	}
	cout << "Total Votes Received: " << total_votes << endl;

	//If total votes (including abstain) has reached the maximum number of students
	if (total_votes == MaxStudents)
	{
		//set nominee 1 as the nominee that has the most vote
		highest_vote = nominee[0].vote;
		for (int i = 1; i < 5; i++)
		{
			if (nominee[i].vote == highest_vote)
			{
				//set to 999 if there is a tie
				topNominee = 999;
			}
			else if (nominee[i].vote > highest_vote)
			{
				highest_vote = nominee[i].vote;
				topNominee = i + 1;
			}
		}

		//dicclare the winner
		winnerDeclaration(nominee, topNominee);
	}
```

```
	else
	{
		//Display voting session havent started yet if  total votes (including abstain) havent reached the maximum number of students
		cout << "Winner will be declared once all students have voted!" << endl;
	}
	system("pause");
}
void winnerDeclaration(NomineeType nominee[], int topNominee)
{
	//If there is a tie
	if (topNominee == 999)
	{
		//Display tie meesage
		cout << "-------------------------------------------------------------------------------------------" << endl << endl;
		cout << "There is an event of a tie." << endl;
		cout << "Another voting session will be held again at another time to be  determined by the administrators" << endl << endl;
		cout << "-------------------------------------------------------------------------------------------" << endl << endl;
	}
	else
	{
		//If there is a winner
		//Display winner declaration
		cout << "-------------------------------------------------------------------------------------------" << endl << endl;
		cout << "The winner is candidate " << topNominee << endl << endl;
		cout << left << setw(15) << "MyKad No" << ":" << nominee[topNominee - 1].n_MyKad << endl; // topNominee - 1 means the position of winner in database
		cout << left << setw(15) << "Student ID" << ":" << nominee[topNominee - 1].n_StudentID << endl;
		cout << left << setw(15) << "Program" << ":" << nominee[topNominee - 1].program << endl;
		cout << left << setw(15) << "Year Of Study" << ":" << nominee[topNominee - 1].yearOfStudy << endl;
		cout << "-------------------------------------------------------------------------------------------" << endl << endl;
	}
```

```
	system("pause");
}

void statistics(NomineeType nominee[], StudentType student[])
{
	int totalFemaleVotes = 0, totalMaleVotes = 0, totalvotes = 0;

	cout << "-----------------------------------------------------------------------------------------------" <<
endl << endl;
	cout << "                                Statistics                                " << endl << endl;
	cout << "-----------------------------------------------------------------------------------------------" <<
endl;

	//read result from Nomination file
	readNominationFile(nominee);

	//accumulate votes (does not include abstain)
	for (int i = 0; i < 5; i++)
	{
		totalvotes += nominee[i].vote;
		totalFemaleVotes += nominee[i].femalevote;
		totalMaleVotes += nominee[i].malevote;
	}

	cout << "Total Votes Received: " << totalvotes << endl;

	//abstain
	cout << "Total number of students who did not vote: " << nominee[5].vote << endl;
	//percentage of students voted
	cout << "Percentage of students voted " << fixed << setprecision(2) << (double)totalvotes /
(totalvotes + nominee[5].vote) * 100.0 << "%" << endl;
	//total votes received by each candidate
	for (int i = 0; i < 5; i++)
	{
		cout << "Votes received by candidate " << i + 1 << " : " << nominee[i].vote << endl;
	}
	// breakdown of votes by gender
```

```
	//total female voters
	cout << "Number of female voters:" << totalFemaleVotes << endl;
	//total male voters
	cout << "Number of male voters:" << totalMaleVotes << endl;
	system("pause");



}

void welcomeAdmin()
{
	cout << "-----------------------------------------------------------------------------------------" <<
endl;
	cout << "                          Welcome ,administrator !                          " << endl;
	cout << "-----------------------------------------------------------------------------------------" <<
endl;
}
void farewellStudents()
{


	cout << "-----------------------------------------------------------------------------" << endl;
	cout << "                      Thanks for your voting !                  " << endl;
	cout << "                                                              " << endl;
	cout << "                       / \\            / \\                    " << endl;
	cout << "                      /   \\          /   \\                   " << endl;
	cout << "                                                              " << endl;
	cout << "                                                              " << endl;
	cout << "                      ||||||||||||||||||||||||              " << endl;
	cout << "                                                              " << endl;
	cout << "                        Have a nice day                       " << endl;
	cout << "-----------------------------------------------------------------------------" << endl;
}

void detailedAnalysisScreen(NomineeType nominee[])
{
```

```
	int choice, labelOfNominee, actualNumOfNomineeInFile;
	// actualNumOfNomineeInFile means the position of the nominee in database as the first nominee is in zero index so we have to minus 1 from the label of nominee
	int totalvotes = 0;
	//Menu for detailed analysis
	cout << "-----------------------------------------------------------------------------------------------" << endl;
	cout << "                              DETAILED ANALYSIS FOR EACH NOMINEE                               " << endl;
	cout << "-----------------------------------------------------------------------------------------------" << endl << endl;
	//Several analysis option including exit from this page
	cout << "[1] Percentage of votes for each nominee" << endl;
	cout << "[2] Number of female voters" << endl;
	cout << "[3] Number of male voters" << endl;
	cout << "[4] Exit" << endl;
	cout << "-----------------------------------------------------------------------------------------------" << endl;

	//read result from Nomination file
	readNominationFile(nominee);

	//Calculate total votes received ( excludes abstain)
	for (int i = 0; i < 5; i++)
	{
		totalvotes += nominee[i].vote;
	}

	do
	{
		cout << endl;
		//Prompt and get choice
		cout << left << setw(40) << "Enter your choice" << ": ";
		cin >> choice;
		//If choice is not equal to 4
		if (choice != 4)
		{
			//Prompt and get label of nominee
			cout << left << setw(40) << "Enter the label of candidates" << ": ";
```

```
                cin >> labelOfNominee;
                //While labelOfNominee is not in the range of 1 to 5
                while (labelOfNominee > 5 || labelOfNominee < 1)
                {
                        //Display error message and prompt the labelOfNominee again
                        cout << "Invalid label. Please enter again. " << endl;
                        cout << "\nEnter the label of candidates:";
                        cin >> labelOfNominee;
                }
                //Calculate the actualNumOfNomineeInFile (position of the nominee in database)
                actualNumOfNomineeInFile = labelOfNominee - 1;



                switch (choice)
                {
                case 1:
                {
                        cout << left << setw(36) << "The percentage of votes for nominee" << setw(4) << labelOfNominee << ": " << fixed << setprecision(2) << (nominee[actualNumOfNomineeInFile].vote / static_cast<double>(totalvotes)) * 100.0 << "%" << endl;
                        break;
                }
                case 2:
                {
                        cout << left << setw(36) << "Number of female voters for nominee" << setw(4) << labelOfNominee << ": " << nominee[actualNumOfNomineeInFile].femalevote << endl;
                        break;
                }
                case 3:
                {
                        cout << left << setw(36) << "Number of male voters for nominee" << setw(4) << labelOfNominee << ": " << nominee[actualNumOfNomineeInFile].malevote << endl;
                        break;
                }
                case 4:
                        cout << "You are exiting" << endl;
                        break;
                default:
```

```
                                cout << "Invalid choice" << endl;
                        }
                }
        //Loop again if the choice is not equal to 4
        } while (choice != 4);



}

void readNominationFile(NomineeType nominee[])
{
        ifstream inData;
        inData.open("NominationList.txt");
        for (int i = 0; i < MaxNominees; i++)
        {
                inData >> nominee[i].n_MyKad >> nominee[i].n_StudentID >> nominee[i].program >>
nominee[i].yearOfStudy >> nominee[i].vote >> nominee[i].femalevote >> nominee[i].malevote;


        }
        inData >> nominee[5].vote;
        inData.close();
}
void readStudentFile(StudentType student[])
{
        ifstream inData;
        inData.open("RegisterStudent.txt");
        for (int i = 0; i < MaxStudents; i++)
        {
                inData >> student[i].MyKad >> student[i].StudentID >> student[i].nominateChance >>
student[i].voteChance;


        }
        inData.close();
}
void updateStudentFile(StudentType student[])
{
        ofstream outData;
        outData.open("RegisterStudent.txt");
```

```
	if (!outData)
	{
		cout << "Error opening data file" << endl;
	}
	else
	{
		for (int i = 0; i < MaxStudents; i++)
		{
			outData << student[i].MyKad << " " << student[i].StudentID << " " << student[i].nominateChance << " " << student[i].voteChance << endl;
		}
	}
	outData.close();
}
void updateNominationFile(NomineeType nominee[])
{
	ofstream outData;
	outData.open("NominationList.txt");
	if (!outData)
	{
		cout << "Error opening data file" << endl;
	}
	else
	{
		for (int i = 0; i < MaxNominees; i++)
		{
			outData << nominee[i].n_MyKad << " " << nominee[i].n_StudentID << " " << nominee[i].program << " " << nominee[i].yearOfStudy << " " << nominee[i].vote << " " << nominee[i].femalevote << " " << nominee[i].malevote << endl;
		}
		outData << nominee[5].vote << endl;
	}
	outData.close();
}
int checkDuplicateStudent(StudentType student[], int studentNo)
{
	bool duplicate = false;
	//If the number of students in file is more than zero
```

```
	if (studentNo > 0)
	{
		//read from student file
		ifstream inData;
		inData.open("RegisterStudent.txt");

		if (!inData) {
			cout << "Error opening files.";
		}
		else {
			//Loop every students infront of the new students who is being registered
			for (int i = 0; i < studentNo; i++)
			{
				inData >> student[i].MyKad >> student[i].StudentID >> student[i].nominateChance >> student[i].voteChance;

				//If there is a matching of MyKad and StudentID between input and database
				if (strcmp(student[studentNo].MyKad, student[i].MyKad) == 0 || strcmp(student[studentNo].StudentID, student[i].StudentID) == 0)
				{
					//Set duplicate to true
					duplicate = true;
					//Display error message
					cout << "Student entered is a duplicate. Registration failed." << endl << endl;
					break;
				}
				else
				{
					duplicate = false;
				}
			}

		}
		inData.close();
	}
	else
	{
```

```
                //Set duplicate to false if the number of students in file is not more than zero
                duplicate = false;
        }


        if (duplicate == true)
                return 1;
        else
                return 0;
}
//Read number of students in file
int readNumStudentsFile()
{
        int number_of_students = 0;
        string line;

        ifstream studentfile("RegisterStudent.txt");
        while (getline(studentfile, line))
                ++number_of_students;
        return number_of_students;
}
//Read number of nominees in file
int readNumNomineeFile()
{
        int number_of_nominee = 0;
        string line;

        ifstream nomineefile("NominationList.txt");
        while (getline(nomineefile, line))
                ++number_of_nominee;
        return number_of_nominee;
}
void clearStudentFile()
{
        ofstream outData;
        outData.open("RegisterStudent.txt");
        if (!outData)
```

```
	{
		cout << "Error opening data file" << endl;
	}

	outData.close();
}
```

Lesson learnt

Wong Yee En

I have learnt the way to pass the parameter between functions and how to store data permanently by using text file. Parameter is a really convenient feature to use in a function as it can widen the ability of a function to a certain extent. For example, passing inputs received from user into the function to be used in a calculation. Besides that, I also got the basic knowledge of drawing flowchart which is taught by tutor Miss Yasotha who has guided us throughout the completion of assignments. Flowchart could help me in the future to draw out the flow of my program and assist in the flow of completing my program. Other than that , I realized the importance of writing comment because it is important to let the other teammates know the concept so that they can continue on coding.

Yam Jason

During the progress of completing the assignment, I have learnt a bunch of advanced C++ coding skills. For instance, implementing the file function, using a file as a medium of storage to store students' data and nominees' data. The file function is very convenient as it can backup data easily. Besides that, I have learnt how to code validation for password and such. Validation is a key in programming, learning how validation works will surely help me to become a better programmer. I've never thought about commenting on my code until I have started this assignment. During the discussion with my mates, I have learned how important commenting is, this is all because it can aid others to understand your codes significantly and work together. My flowchart drawing skill has also increased to a certain professional degree, this is all because of my assignment mates and excellent tutor's teachings. All in all, this assignment is surely an absolute aid to help me become a professional programmer.

Tang Wen Zhi

The lesson I learned is the importance of coding logical thinking, how it works, the basic concepts and logic of drawing a flowchart before coding can give us a clear and simple understanding of how the program works. The basics of drawing flowcharts are the necessary basics that should be learned before coding. I think learning pointers is the hardest subject for me. Actually, the concept and examples of pointers are easier to understand, but the logic of pointers is harder for me.

Conclusion

This assignment is truly one of the milestones to help us become professional programmers. We have gained so much from this assignment, such as the ability to write codes efficiently, how to debug codes, and use structure arrays. Teamwork is one of the factors in completing this assignment, all of us have learned how to work together in order to code a successful program and to complete this assignment. We have learned to acknowledge each other's ideas to brainstorm a better idea to help the assignment. The things we have learned while working on this assignment will not only help us in programming but also to work with other people in the future