****

Problem Solving and C Programming

End Sem Project

**Class Representative Voting System**

**Name- Rohit Praveen Nair**

**Introduction**

PROJECT NAME: Class Representative Voting System

* A Voting system which allows students to vote for their C.R.
* Students can register and cast their vote, while the admin can register candidates and display the results.

PURPOSE OF PROJECT: These three points highlight the purpose of a class representative voting system:

1. Gives power to students to select Representatives for their needs and concerns.

2. Encourages active engagement, improves students’ democratic values and participation.

3. Elected C.R facilitates communication and collaboration between students and the class advisor.

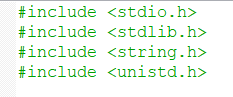
PROBLEM STATEMENT:

“A class representative voting system that manages the process of electing class representatives. The system should allow students to cast their votes, maintain anonymity, prevent fraud activities, and provide an efficient method to count and display the election results accurately. The system should have a user-friendly interface accessible to all eligible students, promoting a fair and transparent election process”

PROGRAMMING LANGUAGE 🡪 C

LIBRARIES USED IN THE PROGRAM:

A library in C is a collection of header files, exposed for use by other programs. The library therefore consists of an interface expressed in a.h file (named the "header") and an implementation expressed in a.



1. *#include <stdio.h>*: A preprocessor directive in the C programming language that is used to include the standard input-output library in a C program. This library contains functions for input and output operations, such as printf and scanf.
2. *#include<stdlib.h>*: The name "stdlib" stands for "standard library". This standard library includes functions involving memory allocation, process control, conversions and others such as system() functions.
3. *#include<string.h>*: A standard header file in the C language that contains functions for manipulating strings (arrays of characters) like strcmp(), strlen(), strcpy(), strcat(), etc.
4. *#include<unistd.h>*: The header defines miscellaneous symbolic constants and types, and declares miscellaneous functions.

We are implementing this header so that we could use sleep()

Function defined in it.



**PROJECT DESCRIPTION**

The C.R Voting System is a program designed to simplify the election process for selecting a Class Representative (C.R). This system enables smooth interaction for administrators and voters, to facilitate essential actions like candidate registration, vote casting, and result display.

FUNCTIONALITIES:

**Candidates**: The program allows administrators to register candidate name and ID associated with the Candidate. The maximum number of candidates that can be registered is defined by the candidates\_max constant.

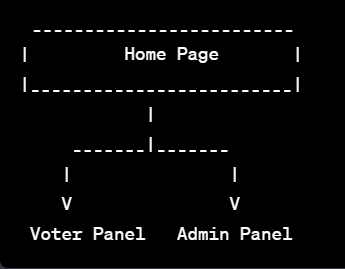
**Voters**: Voters can register by providing their registration number, name, and branch. The program checks if the voter is already registered before allowing them to cast their vote. The maximum number of voters that can be registered is defined by the voters\_max constant.

**Admin Panel**: The Admin Panel can be accessed only if the passkey is correct and it provides options for administrators to register candidates and display the voting results.

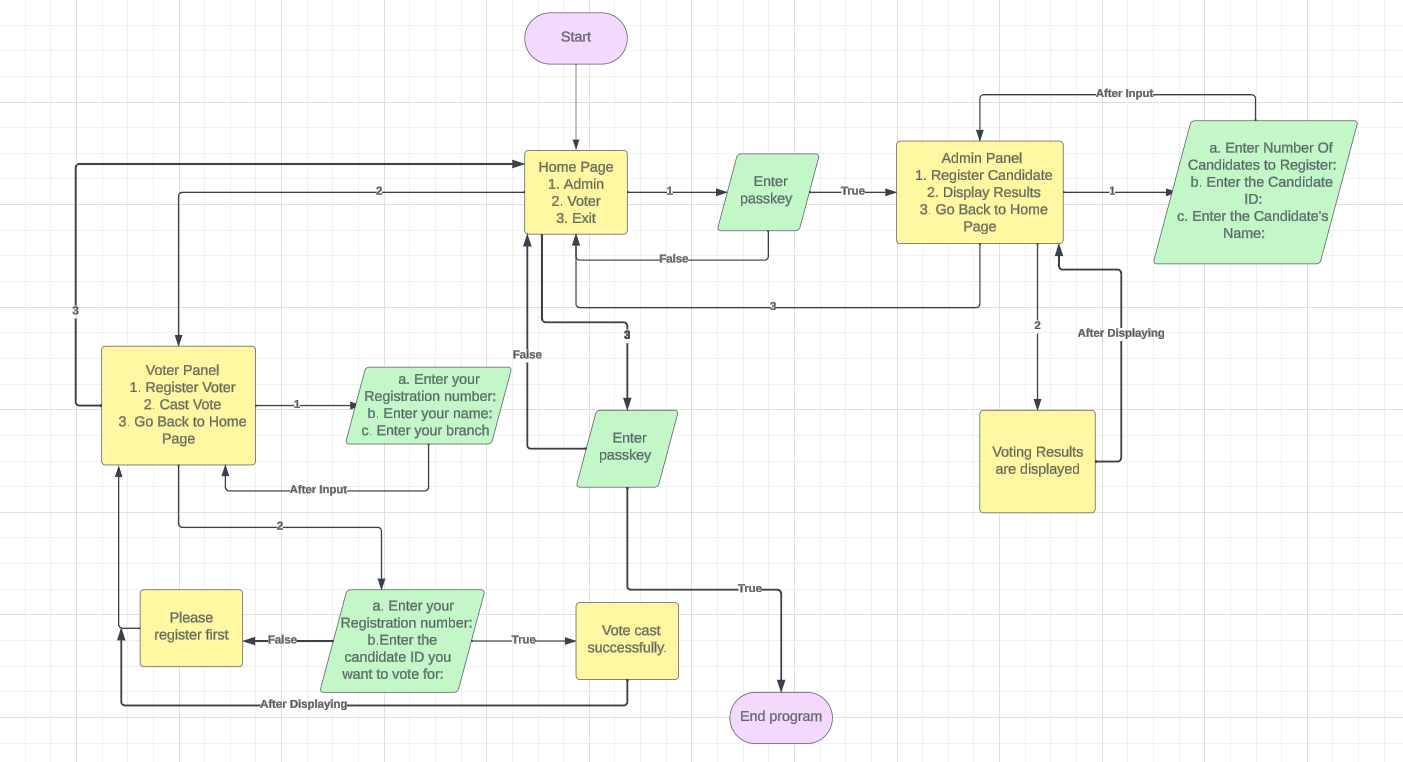
**Voter Panel**: The Voter Panel allows voters to register and cast their votes.

**Home Page**: The Home Page is the starting point of the program and provides options for accessing the admin panel, voter panel, or exiting the program\*.

\*Only the admin has access to exit the program



FLOWCHART:



ALGORITHMS:

An algorithm is a sequence of instructions that are carried out in a sequence in order to solve a problem or complete a work.

Algorithms are often implemented using functions and control structures like loops and conditional statements to carry out specific tasks as defined in it.

The Algorithms used in this program are:

1. **Admin\_Login Algorithm**:

Requires user’s input of the admin passkey and checks if it matches the predefined passkey(“pscp”). Uses strcmp() function to compare the entered passkey with the pre-defined passkey.

This algorithm gives access to AdminPanel module if the entered passkey matches with pre-defined pass key

1. **Candidate\_Register Algorithm**:

Requires the User’s input of number of candidates to register. It uses a for loop to register each candidate by taking input for candidate’s name and the ID associated with the candidate. It checks if the number of candidates registered exceeds the max\_candidates limit.

3) **Display\_Results Algorithm**

It displays the registered candidates along with their respective votes after every voter has finished voting. It checks the candidate with the highest votes or checks for a tie. The candidate with highest number of votes is displayed as the Class Representative.

4) **Voter\_Register Algorithm**

Registers a voter by taking input for registration number, name, and branch. It also checks if the voter has already registered before in order to prevent the same voter to vote again. Prevents registration of voter if the maximum number of votes has crossed.

5) **Vote\_Cast Algorithm**

Allows a registered voter to cast their vote by choosing a candidate ID. It Checks if the voter has registered and hasn't voted yet. If the voter isn’t registered then it asks user to register before casting vote. It records the vote for the chosen candidate or considers it as NOTA (None of The Above) if the candidate ID is invalid.

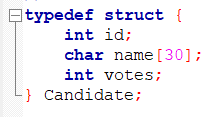
DATA STRUCTURES:

Two data structures have been implemented in this program:

1) ***struct Candidate****;*

This struct is used to store data of candidates like storing their ID, name, and number of votes in an array named candidates.

Typedef command is used to rename structures so that we can refer structure variables easily.



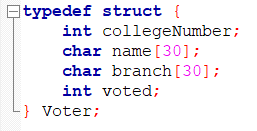
*int id*: Stores the Candidate ID inputted by admin.

*char name [30]*: Stores Candidate name (maximum 30 characters).

*Int votes*: Stores the number of votes received by the candidate.

2**) *struct Voter****;*

This struct is used to store data of voters like storing their registration number, name, branch, and a flag variable to indicate if they have voted or not in an array named voters.



Typedef command is used to rename structures so that we can refer structure variables easily.

*int collegeNumber*: Stores Registration number of the Voter.

*char name*: Stores Name of the voter (maximum 30 characters).

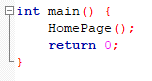
*char branch*: Stores branch of the voter (maximum 30 characters)

*int voted*: Indicates whether the voter has casted their vote (0 for not voted,1 for voted).

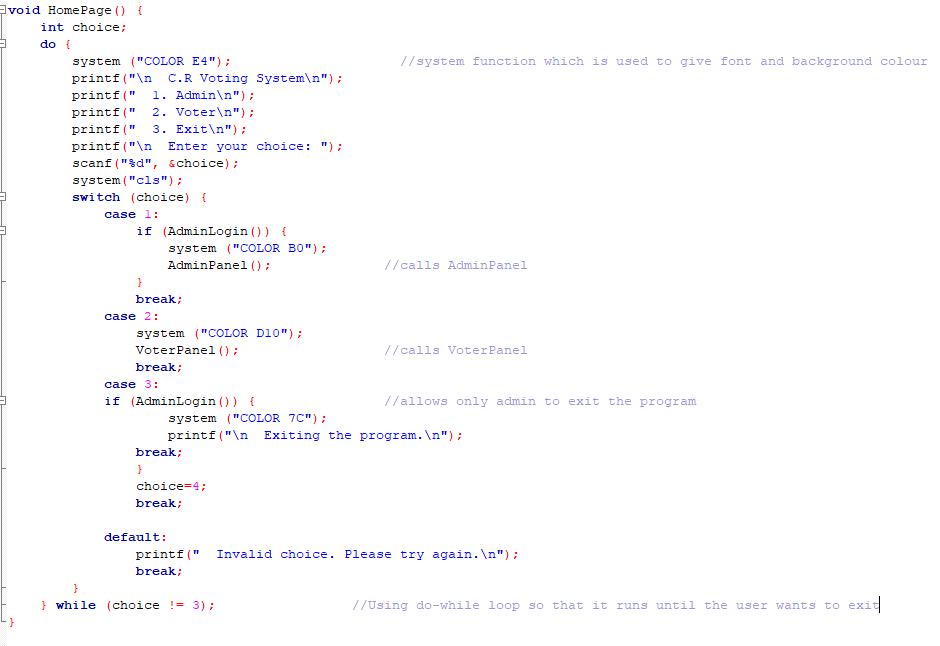
**IMPLEMENTATION**

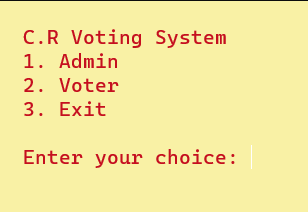
Step-By-Step Working

1) **The Main Function:** It calls HomePage() function



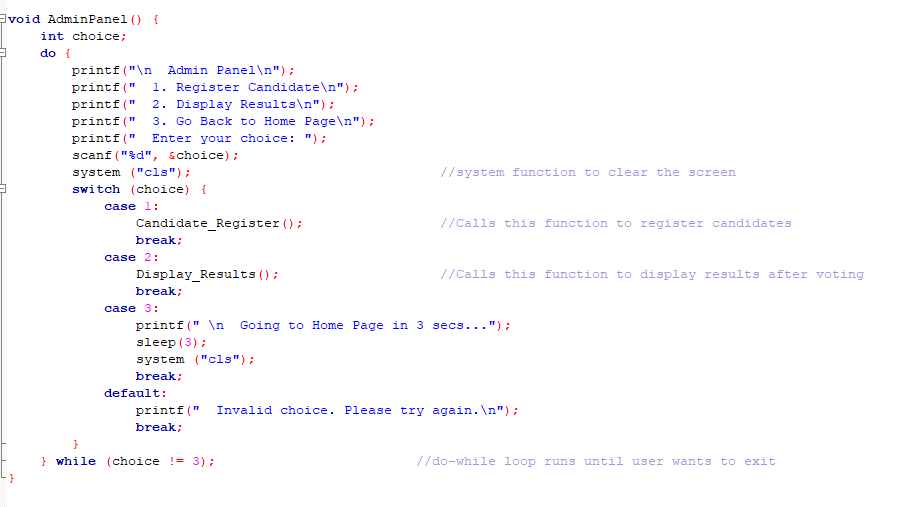
2) **Home Page:**

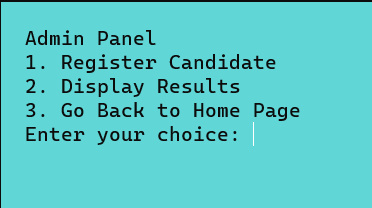


Output: 

* In the HomePage() function there is a menu driven switch case approach where the different options are firstly printed using printf() and the choice entered by the user is stored in the variable choice
* This choice variable is passed to switch case, the case matching the value of choice variable is executed further and there is a break statement after every case to ensure that once the case is executed the next cases are omitted.
* Case 1: Takes the control to AdminLogin() function which checks if the inputted passkey is matching with pre-defined passkey and if it returns 1 then it changes the font and background colour and takes control to AdminPanel() function.
* Case2: Takes Control to VoterPanel() function and changes font and background colour according to the colour code given.
* Case 3: Takes control to AdminLogin() function to check if passkey entered is correct and exits the program. Thus it gives power to the admin to exit the program.

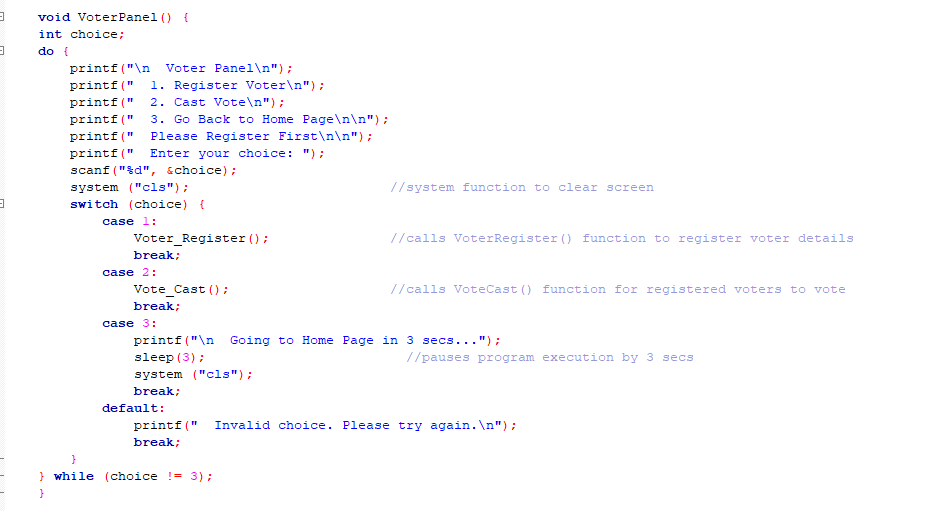
3) **Admin Panel:**

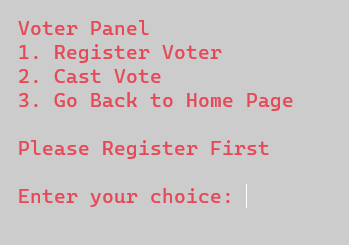


Output: 

* Admin Panel Function is a basic menu-driven switch case approach where the choice entered by the admin is stored in choice variable. The choice variable is passed on to switch case and the respective case which is matching the choice variable is executed.
* Case 1: It takes the control to Candidate\_Register() function whose basic functions are to register candidate name and ID associated with the candidate.
* Case 2: It takes the control to Display\_Results() function whose basic functions are to display the candidate with most votes as the C.R after every voter has voted.
* Case 3: It exits the switch case by pausing the execution by the number of seconds defined in the sleep function (in this case 3 secs).

4) **Voter Panel:**

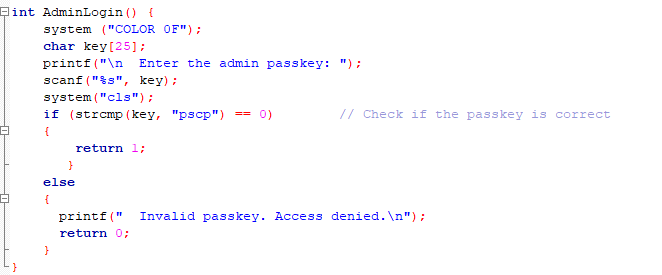


Output: 

* Voter Panel Function is a basic menu-driven switch case approach where the choice entered by the voter is stored in choice variable. The choice variable is passed on to switch case and the respective case which is matching the choice variable is executed.
* Case 1: It takes the control to Voter\_Register() function whose basic functions are to register voter’s college Registration number, name and branch and it prevents an already registered voter to register again.
* Case 2: It takes the control to Vote\_cast() function which allows registered voter to select candidate ID associated with the candidate and cast their vote and it prevents a voter to vote 2 times and doesn’t allow the voter to vote without registering.
* Case 3: It exits the switch case by pausing the execution by the number of seconds defined in the sleep function (in this case 3 secs).

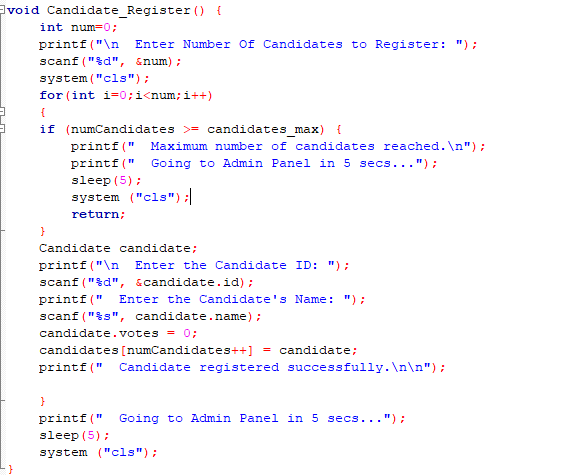
**Functionalities in the program:**

1) **Admin Login Function**: Checks if the passkey inputted by the admin is matching with the pre-defined passkey and returns 1 or 0 if its matching or not respectively.



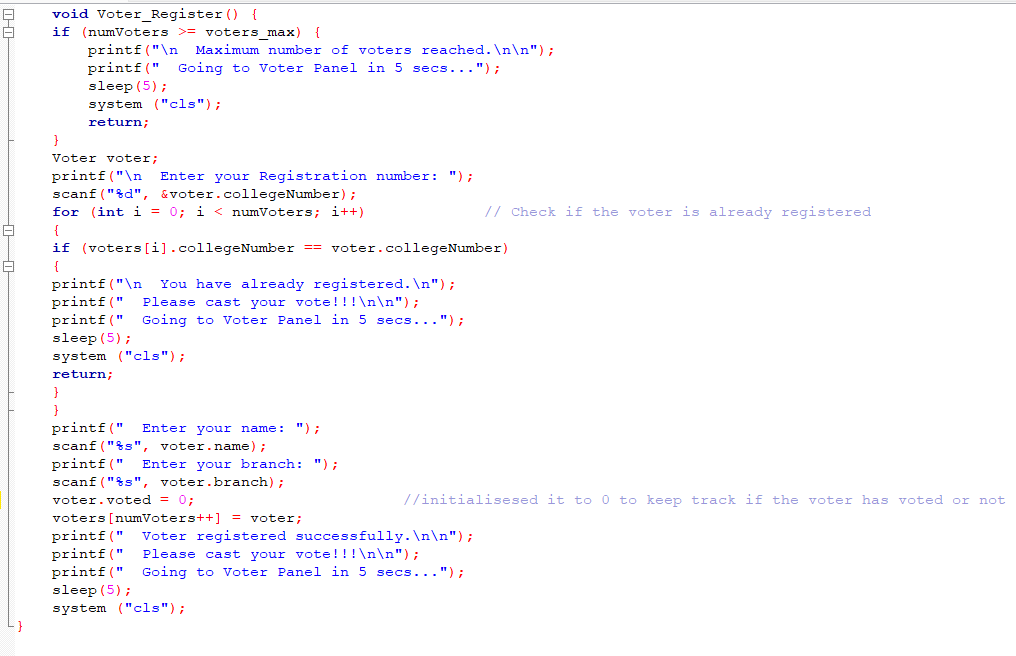
* AdminLogin() function takes admin’s input and stores it in variable key of data type char (maximum 25 characters) and using strcmp() which is a string function is used to compare 2 strings.
* Strcmp() compares 2 strings and returns 0 if there are equal , It returns +ve number if 1st string is greater than 2nd string and returns -ve number if 1st string is lower than 2nd string lexographically.
* It compares if entered string is equal to “pscp” and if it is equal to 0 then it means it is equal and it returns 1 else it returns 0.

2) **Candidate Register Function:**  Admin needs to input the number of candidates to register and then requires admin to input name and ID of the candidate.



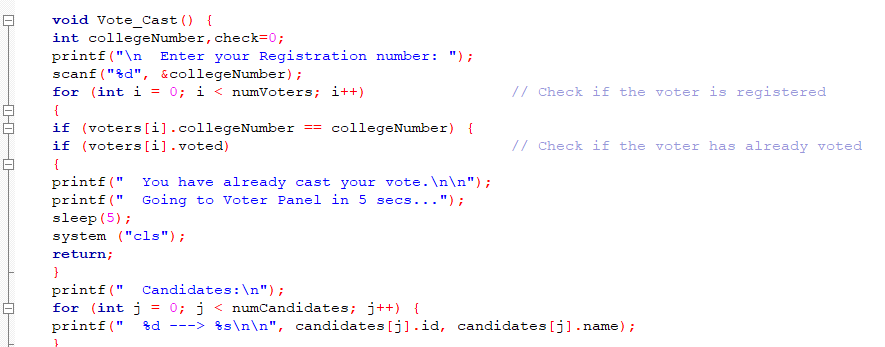
* Candidate\_Register () function stores the number of candidates to be registered in variable num and then clears screen.
* It uses for loop to check if the total number of candidates registered exceeds the defined macro candidates\_max (which is 5 in this case). Macro is a name given to a piece of code and whenever the compiler encounters a macro, it will replace it with the macro value.
* Now using arrays of structure concept, we can store candidates ID and name in array candidate of structure Candidate.
* We initialised candidate.votes=0 for the candidate in order to count the number of the candidate gets during voting.
* After registering one candidate we increment numCandidates by 1 and store it in an array candidates and display the message candidate registered successfully.
* This loop runs till it reaches the number of candidates to be registered but can’t exceed 5 and it pauses the program execution for 5 secs and clears screen.

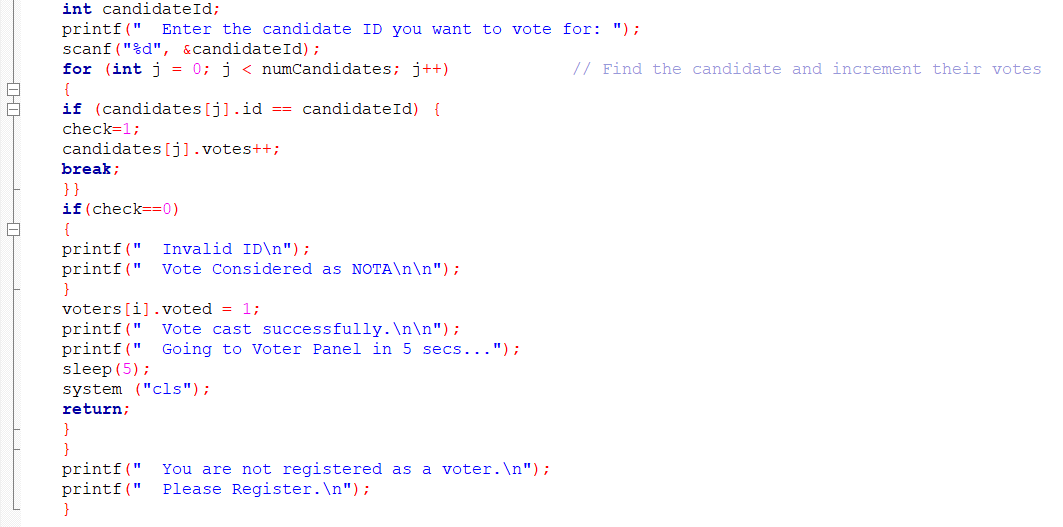
3) **Voter Registration Function**: Voter needs to input their college Registration number and it checks if you have already registered before in order to prevent same person to vote twice. If not registered it will ask for Voter name and branch and stores the value in an array of structures.



* Voter\_Register function first checks if number of voters registered exceeds the defined macro voters\_max (which is 70 in this case), if it exceeds 70 then it displays maximum number of voters reached and takes control back to Voter Panel.
* It then stores the voters registration number, name and branch in an array of structures and it initialises voter.voted to 0 in order to keep track if voter has voted or not.
* It then increments numVoters by 1 and stores the structure in that array and displays message voter registered successfully.

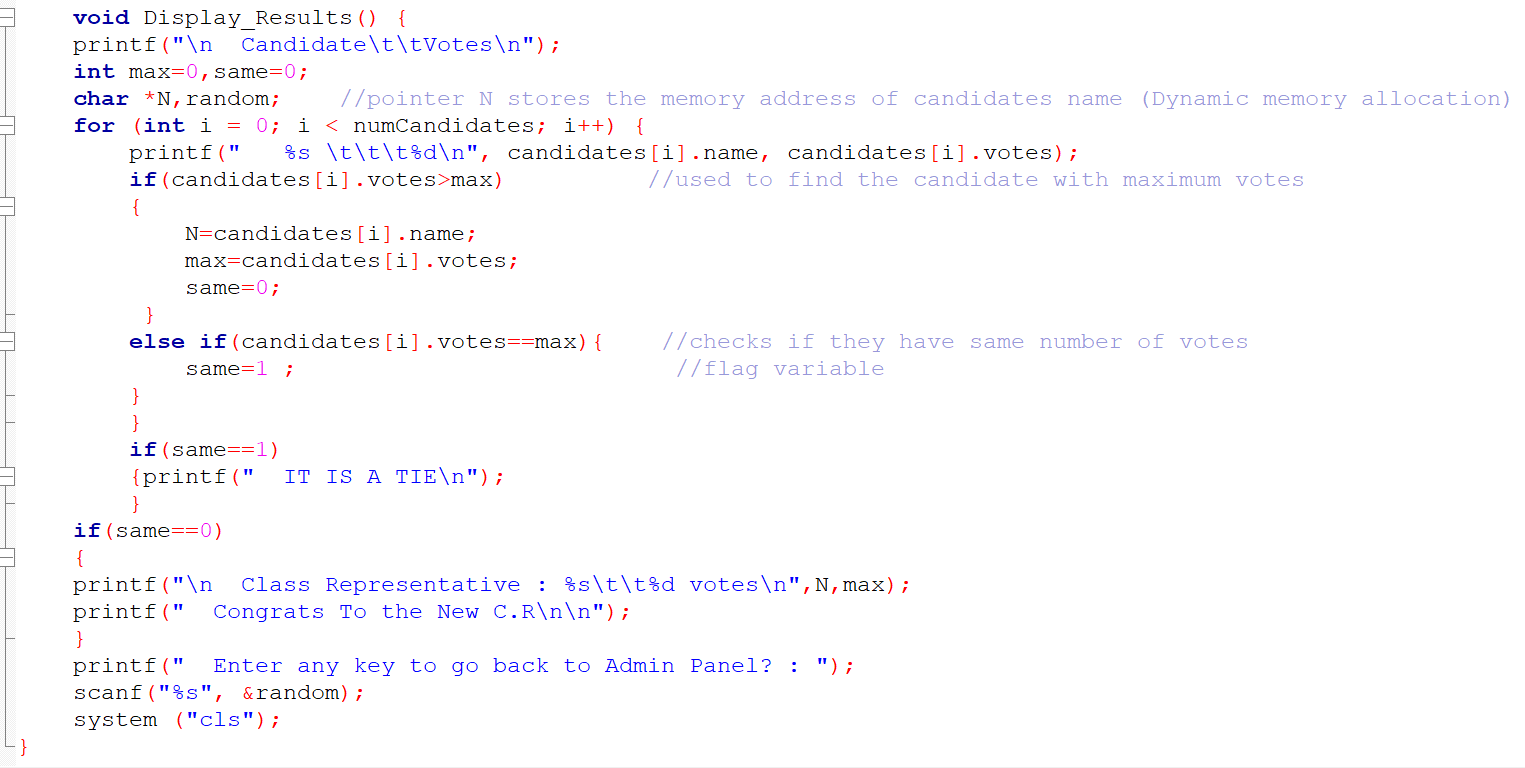
4) **Cast Vote Function**: This function allows registered voters to vote for their candidate and it prevents fraud actions like an already registered voter can’t vote again and a voter can’t vote twice.





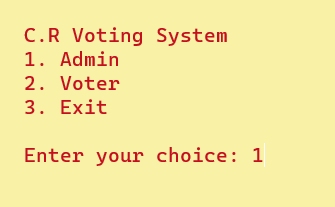
* Vote\_Cast function first stores the voters registration number in variable collegeNumber and checks that variable if its matching with any registered voter’s registration number in the array voters by utilising for loop in order to find out if the voter has registered or not and to prevent that voter to vote twice.
* If the variable collegeNumber is matching that means its registered, now it uses an if condition in it to check if the person has already voted or not.
* If Voters[i].voted==1 then it means the voter has voted and it displays “You have already casted your vote” and exits the function.
* If Voters[i].voted==0 then it means voter hasn’t casted vote and it then displays the candidate’s ID and name stored in the array candidates by utilising a for loop.
* The function then asks the voter input the candidate ID they choose to vote for and stores it in variable candidateID and uses a for loop to search for that specific candidate ID stored in array candidates.
* We use a flag variable *check* to see If the candidate ID matches and if it does then check=1 and the votes of that candidate is incremented by 1 and stored in that array candidates, but if the ID doesn’t match then check=0 and the vote is taken as NOTA and voter is considered to have voted.
* After casting the vote, then Voters[i].voted=1 which states that the voter has voted and displays message “vote cast successfully” and clears screen.

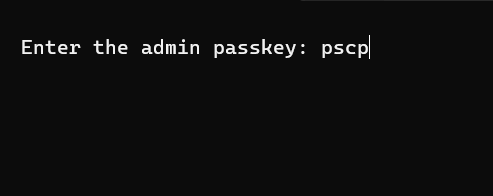
5) **Display Result Function**: This function is used to display the results of the voting session by finding the candidate with maximum number of votes and displaying him/her as the C.R or in case of same number of votes will be displayed as a tie.

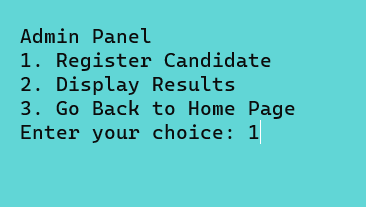


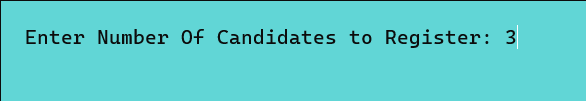
* This function displays the candidate names along with the number of votes they received with the help of a for loop.
* In order to find the name of the candidate with highest number of votes we use pointers (char \*N) cause this is functioning in a for loop and its efficient to store the memory address of the name so that it dynamically allocates the memory value to N.
* We use flag variable *same* to check if it’s a tie or not.
* If same==0 then the class representative name and total votes is displayed and the purpose of this entire program is served.

**TESTING AND RESULTS**

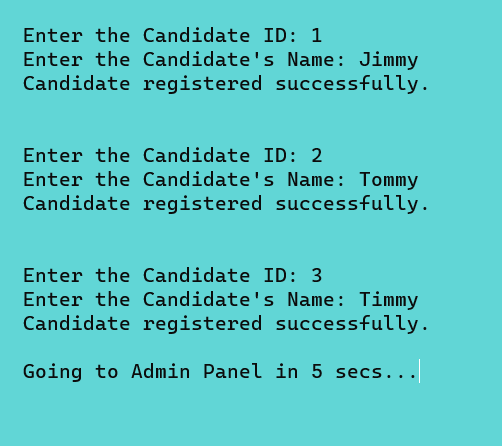


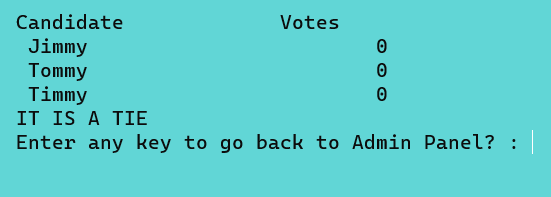




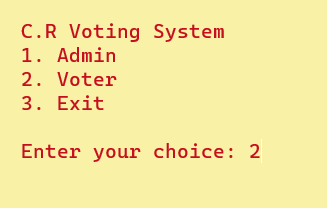


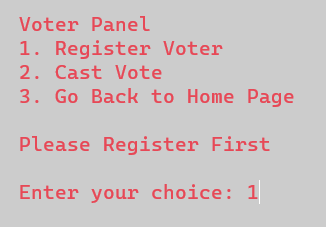
(Can’t cross more than 5)



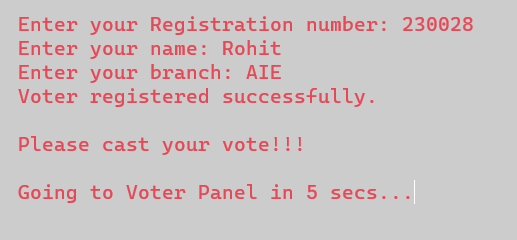


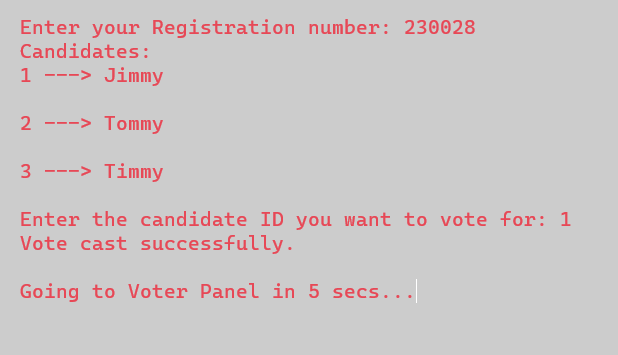
(To show all the votes have been initialised to 0)

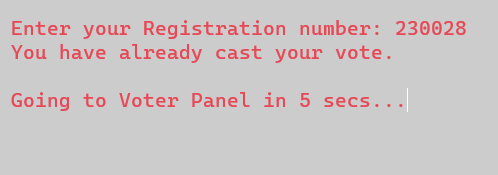




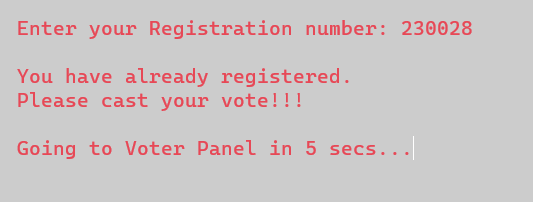
(This testing is shown only for one voter)



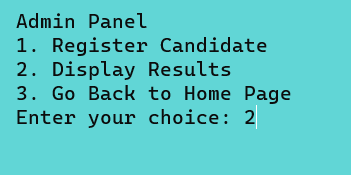


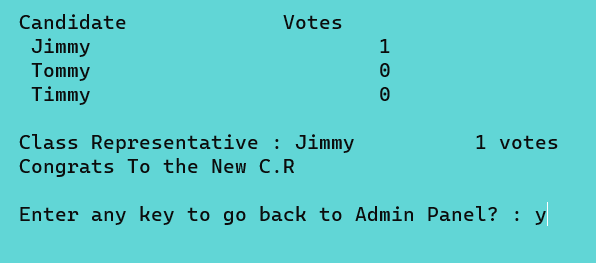


(Doesn’t allow same person to vote twice)

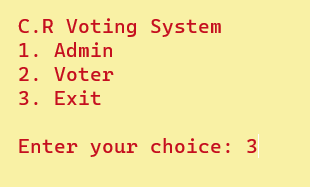


(Doesn’t allow same to register twice)

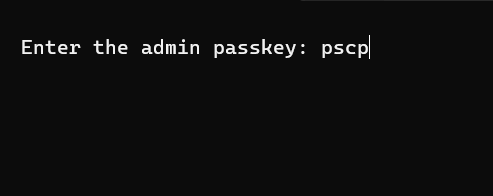


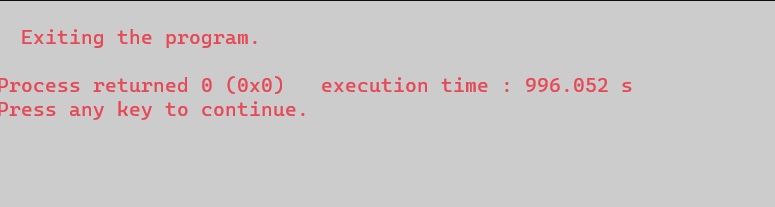


(Displays the results and its usually done after every student has voted)



(Only admin has to access to end the program which prevents voters to cause chaos to the system by ending it)





**BUGS:**

BUG 1:

The program encountered bug when the admin entered wrong password and it instead of taking control back to Home Page, it just exited the program.

FIX:

We corrected the if condition where it is uses strcmp to compare 2 strings and added an else condition to return 0 along with a message.

BUG 2:

When all the candidates had same number of votes, the program encountered a logical bug and displayed the first candidate as the C.R.

FIX:

We made use of a flag variable *same* and initialised it to 0 when we found one candidate with maximum number of votes and initialised it to 1 when they had same number of votes.

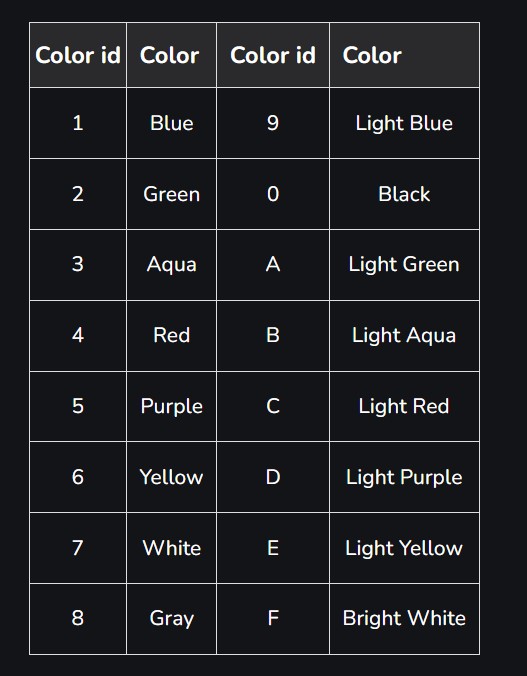
Then implemented a nested if condition to display the message “IT IS A TIE” when check==1 and displayed the C.R along with the votes when check==0.

BUG 3:

The program encountered syntax error while compiling the code. The syntax for a system function was wrong. It gave error for system(“colour,0F”);

FIX:

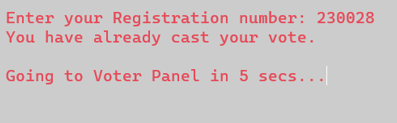
This system function which helped change the background and font colour was implemented to improve the user interface and make it more engaging. The correct line was system(“COLOR,0F”);



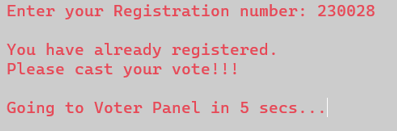
*Colour code*

**ERROR HANDLING**:

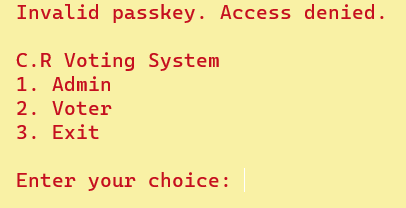
These outputs show that the program is capable of error-handling



(Doest let voter to vote twice)



(Doesn’t let voter to register twice)



(When the admin types wrong passkey)

**CONCLUSION**

**KEY ACHIEVEMENTS:**

1) **Modular Code Structure:** The code is broken down into various functions, each responsible for a specific task which enhances the readability and maintainability.

2) **User-Friendly Experience**: Used many features like clearing the screen, changing bg and font colours for better visual representation and adding delay pauses to improve user experience.

3) **Dynamic use of Arrays**: The Program uses arrays of structures (‘candidates’ and ‘voters’) to manage candidate and voter information dynamically.

**LEARNINGS:**

1) Gained knowledge about data organisation using structures and using arrays of structures.

2) Learnt so much about user interaction and creating an interactive user-interface by using different libraries which expanded our knowledge and perspective of C programming.

**LIMITATION AND FUTURE IMPROVEMENTS:**

When the program finishes execution and terminates then all the voter and candidate stored in memory (in form of arrays) gets lost and there is no method or mechanism to store this data permanently like in a database. Thus, the data is only available during program’s runtime.

This limitation could be improved by storing the candidates and voters’ data in files or database to maintain data even after program termination.