Progress #9

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to votecandidate.c programed by Samuela Abigail- 71762108039 \*

\* \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "votecandidate.h"

/\*\*\*\*\*global variables\*\*\*\*\*\*\*/

//for candidates

struct candidate

{

char cand\_id[45];//candidate ID

char cand\_name[45];//candidate name

char cand\_par[45];//candidate party name

int cand\_div;//candidate division number (division 1 to division 4)

int count;//number of votes of candidate

}cand;

//for voters

struct voters

{

char voter\_id[45];//voter ID

char voter\_name[45];//voter first name

char status;//Y means already voted, N means not voted

int voter\_div;//voter division number (division 1 to division 4)

int age;//age of voter should be >=18

}voter;

/\*\*\*\*\*End of global variables\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*Accessory functions\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear()

{

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function which displays real/actual time

void real\_clock()

{

time\_t t;//for showing actual time

time(&t);//I used time to make this machine look pretty, although not sure if real EVMs display time

//made this much space characters so that clock will be placed in right most corner

green();

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" %s",ctime(&t));//displays time

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of real\_clock() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*End of accessory functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*/

//displays list of all candidates

void full\_candidate\_list()

{

int i=0;

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("D:\\candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\t\t\tDIVISION\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

//https://fresh2refresh.com/c-programming/c-strings/c-strtok-function/

//https://stackoverflow.com/questions/3889992/how-does-strtok-split-the-string-into-tokens-in-c

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

printf("\n%s\t\t\t%s\t%s\t\t\t%s",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_div);

}

i++;

}

fclose(fptr);

//After while loop gets executed, program terminates abruptly

//Don't know why

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of full\_candidate\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list division-wise

void div\_list()

{

int i=0;

char division[2]="2";//for the sake of making this code work, I'm initializing it as string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("D:\\candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter division number: ");

scanf("%s",division );

purple();

printf("\nCandidates from division %s are: \n",division);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

if(strcmp(Cand[i].Cand\_div,division)==0)

{

printf("\n%s\t\t\t%s\t%s",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par);

}

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

}

i++;

}

fclose(fptr);

//After while loop gets executed, program terminates abruptly

//Don't know why

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of div\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list party-wise

void par\_list()

{

int i=0;

char party[100];//for the sake of making this code work, I'm initializing it as string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("D:\\candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter party name: ");

getchar();

scanf("%[^\n]s",party);

purple();

printf("\nCandidates from party %s are: \n",party);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tDIVISION\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

if(strcmp(Cand[i].Cand\_par,party)==0)

{

printf("\n%s\t\t\t%s\t\t\t%s",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_div);

}

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

}

i++;

}

fclose(fptr);

//After while loop gets executed, program terminates abruptly

//Don't know why

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of par\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for viewing candidates in option 1 in menu

void view\_candidates()

{

int option, division;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to List of Election Candidates\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\t1. View all candidates\n");

printf("\t2. View candidates by division\n");

printf("\t3. View candidates by party\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("Enter your choice: ");

scanf("%d",&option);

green();

switch (option) {

case 1:

screen\_clear();

full\_candidate\_list();

break;

case 2:

screen\_clear();

div\_list();

break;

case 3:

screen\_clear();

par\_list();

break;

default:

red();

printf("\nInvalid selection! Please try again.\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of view\_candidates() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*/

//to register as voter

void register\_voter()

{

char temp[100], line[100], str[]="@EVM#";

int i=0,j=0,len,k, dup\_flag=0, no\_space\_flag=1;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Voter Registration Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

for(k=1;k<2;k++ )//for the sake of having a loop to apply break statement.

//We have specified conditions so that for loop will be executed only once,

//otherwise it'll be infinite loop if left as for ( ; ; )

{

cyan();

printf("\*You should be at least 18 years old to register as voter.\n");

printf("\*Your division number should be from 1 to 4.\n");

printf("\*While entering name, please leave space in case of initials. Example- S Sakthisree\n\n\n");

yellow();

printf("\nEnter your full name: ");

getchar();//if not used means scanf won't work for strings involving space characters

scanf("%[^\n]s", voter.voter\_name);

FILE \*file=(fopen("D:\\voter\_list.txt","r"));//opening file in reading mode

if(file==NULL){

red();

printf("\nError! Couldn't open file\n\n");

exit(1);

}

while ( fgets ( line, sizeof(line), file )!=NULL)//checking each line for duplicate voter name

{

char \*ptr = strstr(line, voter.voter\_name);

if (ptr != NULL)

{

dup\_flag=1;//voter name is already registered

}

}

fclose(file);

if(dup\_flag)//breaks out of loop if flag is set to 1

{

red();

printf("Already registered as voter!\n\n");

break;

}

printf("\nEnter your age: ");

scanf("%d", &voter.age);

if(voter.age <18)

{

red();

printf("\nYou are not eligible to be vote!\n\n");

break;

}

printf("\nEnter your division number: ");

scanf("%d", &voter.voter\_div);

if(voter.voter\_div <1 || voter.voter\_div >4)

{

red();

printf("\nInvalid division number!\n\n");

break;

}

green();

printf("\n\nYou have successfully registered as a voter!");

printf("\nHere is your voter ID, PLEASE keep it carefully: ");

len=strlen(voter.voter\_name);

strcpy(temp, voter.voter\_name);//storing voter name in temp.

//If voter.voter\_name itself used means concatenated name will be saved in file in place of actual name

for(i=0,j=0 ; i<len ; i++)//for removing space characters, but won't work for names without space characters,

//and hence their voter ID's will be blank

{

if(temp[i] == ' ' && temp[i]!=NULL)

{

for(j=i ; j<len ; j++)

{

temp[j] = temp[j+1];

}

len--;

no\_space\_flag=0;//flag is set to 0 since voter name has space character

//if voter name doesn't have space character (example- Sasha),it won't go inside above loop and voter ID won't be generated

}

}

temp[j]='\0';//assigning NULL at end of string

if(no\_space\_flag)//generating voter ID for names without space character

{

strcpy(temp,voter.voter\_name);//if I remove this here despite already applying it above for loop, it's not working

strcat(temp, str);//concatenating

strcpy(voter.voter\_id,temp);//voter ID is voter name only with @EVM# in end

}

else//for voters having space character in their name

{

strcpy(voter.voter\_id,temp);//assigning voter ID

}

purple();

printf("%s\n\n", voter.voter\_id);

voter.status='N';

FILE \*fptr=(fopen("D:\\voter\_list.txt","a"));//opening file in appending mode

if(fptr==NULL){

red();

printf("\nError! Couldn't open file\n\n");

exit(1);

}

//storing full name, voter ID, age, division, and status of voter in file

fprintf(fptr,"\n%s\t%s\t%d\t%d\t%c", voter.voter\_name, voter.voter\_id, voter.age, voter.voter\_div, voter.status);

fclose(fptr);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of register\_voter() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*Function to be called inside main() function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void voter\_menu()

{

int option;//used for selecting menu option

blue();

printf("\n\*\*\*\*\*\*\*\*\*WELCOME to Tamil Nadu State Electoral Commission E-system\*\*\*\*\*\*\*\*\*\*");

real\_clock();//calling to display time

cyan();

printf("\t1. View Candidates\n");

printf("\t2. Register Voter\n");

printf("\t3. Vote\n");

printf("\t4. View Voting Results and Summary\n");

printf("\t5. Quit\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("\nEnter your choice: ");

scanf("%d", &option);

switch (option) {

case 1:

view\_candidates();

break;

case 2:

register\_voter();

break;

case 3:

break;

case 4:

break;

case 5:

screen\_clear();

green();

printf("\nThank you for visiting. Goodbye!\n\n");

exit(0);

break;

default:

red();

printf("\nInvalid selection, please try again!\n\n");

}

}

Progress #10

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to votecandidate.c programed by Samuela Abigail- 71762108039 \*

\* \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "votecandidate.h"

/\*\*\*\*\*global variables\*\*\*\*\*\*\*/

//for candidates

struct candidate

{

char cand\_id[45];//candidate ID

char cand\_name[45];//candidate name

char cand\_par[45];//candidate party name

int cand\_div;//candidate division number (division 1 to division 4)

int count;//number of votes of candidate

}cand;

//for voters

struct voters

{

char voter\_id[45];//voter ID

char voter\_name[45];//voter first name

char status;//Y means already voted, N means not voted

int voter\_div;//voter division number (division 1 to division 4)

int age;//age of voter should be >=18

}voter;

/\*\*\*\*\*End of global variables\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*Accessory functions\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear()

{

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function which displays real/actual time

void real\_clock()

{

time\_t t;//for showing actual time

time(&t);//I used time to make this machine look pretty, although not sure if real EVMs display time

//made this much space characters so that clock will be placed in right most corner

green();

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" %s",ctime(&t));//displays time

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of real\_clock() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*End of accessory functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*/

//displays list of all candidates

void full\_candidate\_list()

{

int i=0;

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

//inside candidate list file, if I store details having space characters inbetween, strtok is considering that also as part of string

//so strcmp will give non zero value when comparing in if loop

//so don't leave space between each detail like name and party name

if ((fptr = fopen("D:\\candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\t\t\tDIVISION\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

//https://fresh2refresh.com/c-programming/c-strings/c-strtok-function/

//https://stackoverflow.com/questions/3889992/how-does-strtok-split-the-string-into-tokens-in-c

//I tried using fscanf() with %[^|]s and all, but it didn't work

//Maybe I should have placed integers and strings alternatively in txt files and used fread properly

//anyway that didn't work correctly

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

printf("\n%s\t\t\t%s\t\t%s\t\t\t%s",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_div);

}

i++;

}

printf("\n\n");

fclose(fptr);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of full\_candidate\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list division-wise

void div\_list()

{

int i=0, div\_flag=1;

char division[2]="2";//I'm initializing it as string since it is part of line in file which is a string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("D:\\candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter division number: ");

scanf("%s",division );

purple();

printf("\nCandidates from division %s are: \n",division);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

if(strcmp(Cand[i].Cand\_div,division)==0)

{

printf("\n%s\t\t\t%s\t\t%s",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par);

div\_flag=0;

}

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

}

i++;

}

if(div\_flag)

{

red();

printf("\nEnter correct division number!");

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of div\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list party-wise

void par\_list()

{

int i=0, par\_flag=1;

char party[100];//for the sake of making this code work, I'm initializing it as string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("D:\\candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter party name: ");

getchar();

scanf("%[^\n]s",party);

purple();

printf("\nCandidates from party %s are: \n",party);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tDIVISION\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

if(strcmp(Cand[i].Cand\_par,party)==0)

{

printf("\n%s\t\t\t%s\t\t\t%s",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_div);

par\_flag=0;

}

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

}

i++;

}

if(par\_flag)

{

red();

printf("\nParty not found!");

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of par\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for viewing candidates in option 1 in menu

void view\_candidates()

{

int option;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to List of Election Candidates\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\t1. View all candidates\n");

printf("\t2. View candidates by division\n");

printf("\t3. View candidates by party\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("Enter your choice: ");

scanf("%d",&option);

green();

switch (option) {

case 1:

screen\_clear();

full\_candidate\_list();

break;

case 2:

screen\_clear();

div\_list();

break;

case 3:

screen\_clear();

par\_list();

break;

default:

red();

printf("\nInvalid selection! Please try again.\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of view\_candidates() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*/

//to register as voter

void register\_voter()

{

char temp[100], line[100], str[]="@EVM#";

int i=0,j=0,len,k, dup\_flag=0, no\_space\_flag=1;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Voter Registration Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

for(k=1;k<2;k++ )//for the sake of having a loop to apply break statement.

//We have specified conditions so that for loop will be executed only once,

//otherwise it'll be infinite loop if left as for ( ; ; )

{

cyan();

printf("\*You should be at least 18 years old to register as voter.\n");

printf("\*Your division number should be from 1 to 4.\n");

printf("\*While entering name, please leave space in case of initials. Example- L Sasha\n\n\n");

yellow();

printf("\nEnter your full name: ");

getchar();//if not used means scanf won't work for strings involving space characters

scanf("%[^\n]s", voter.voter\_name);

FILE \*file=(fopen("D:\\voter\_list.txt","r"));//opening file in reading mode

if(file==NULL){

red();

printf("\nError! Couldn't open file\n\n");

exit(1);

}

while ( fgets ( line, sizeof(line), file )!=NULL)//checking each line for duplicate voter name

{

char \*ptr = strstr(line, voter.voter\_name);

if (ptr != NULL)

{

dup\_flag=1;//voter name is already registered

break;//so no need to check voter list further

}

}

fclose(file);

if(dup\_flag)//breaks out of loop if flag is set to 1

{

red();

printf("Already registered as voter!\n\n");

break;

}

printf("\nEnter your age: ");

scanf("%d", &voter.age);

if(voter.age <18)

{

red();

printf("\nYou are not eligible to be vote!\n\n");

break;

}

printf("\nEnter your division number: ");

scanf("%d", &voter.voter\_div);

if(voter.voter\_div <1 || voter.voter\_div >4)

{

red();

printf("\nInvalid division number!\n\n");

break;

}

green();

printf("\n\nYou have successfully registered as a voter!");

printf("\nHere is your voter ID, PLEASE keep it carefully: ");

len=strlen(voter.voter\_name);

strcpy(temp, voter.voter\_name);//storing voter name in temp.

//If voter.voter\_name itself used means concatenated name will be saved in file in place of actual name

for(i=0,j=0 ; i<len ; i++)//for removing space characters, but won't work for names without space characters,

//and hence their voter ID's will be blank

{

if(temp[i] == ' ' && temp[i]!=NULL)

{

for(j=i ; j<len ; j++)

{

temp[j] = temp[j+1];

}

len--;

no\_space\_flag=0;//flag is set to 0 since voter name has space character

//if voter name doesn't have space character (example- Sasha),it won't go inside above loop and voter ID won't be generated

}

}

temp[j]='\0';//assigning NULL at end of string

if(no\_space\_flag)//generating voter ID for names without space character

{

strcpy(temp,voter.voter\_name);//if I remove this here despite already applying it above for loop, it's not working

strcat(temp, str);//concatenating

strcpy(voter.voter\_id,temp);//voter ID is voter name only with @EVM# in end

}

else//for voters having space character in their name

{

strcpy(voter.voter\_id,temp);//assigning voter ID

}

purple();

printf("%s\n\n", voter.voter\_id);

voter.status='N';

FILE \*fptr=(fopen("D:\\voter\_list.txt","a"));//opening file in appending mode

if(fptr==NULL){

red();

printf("\nError! Couldn't open file\n\n");

exit(1);

}

//storing full name, voter ID, age, division, and status of voter in file

fprintf(fptr,"\n%s|%s|%d|%d|%c;", voter.voter\_name, voter.voter\_id, voter.age, voter.voter\_div, voter.status);

fclose(fptr);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of register\_voter() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 3 functions\*\*\*\*\*\*\*\*\*\*\*/

//registered voters will vote

void voting()

{

char buf[100], line[100], id[100];

int i=0,j=0, k, vote\_flag=1;

char\* token;

struct Voter

{

char V\_id[45];//voter ID

char V\_name[45];//voter name

int V\_div;//voter division number (division 1 to division 4)

int V\_age;//voter age

int V\_status;//Y/N

}V[100];

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Election Voting Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

yellow();

printf("\nEnter your voter ID: ");

scanf("%s", id);

FILE \*file=(fopen("D:\\voter\_list.txt","r"));//opening file in reading mode

if(file==NULL){

red();

printf("\nError! Couldn't open file\n\n");

exit(1);

}

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), file)){

if(fscanf(file, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");//content of line till 1st '|' is encountered gets stored in token now

if(feof(file))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(V[i].V\_name,token);

printf("%s",V[i].V\_name);

token = strtok(NULL, "|");

strcpy(V[i].V\_id ,token);

printf("\n%s\n",V[i].V\_id);

token = strtok(NULL, "|");

V[i].V\_age=token;

printf("%s\n",V[i].V\_age);

token = strtok(NULL, "|");

V[i].V\_div = token;

printf("%s\n",V[i].V\_div);

token = strtok(NULL, "|");

V[i].V\_status=token;

printf("%s\n",V[i].V\_status);

token = strtok(NULL, "|");

}

if(strcmp(V[i].V\_id, id)==0 && strcmp(V[i].V\_status,"N")==0)

//though status is a single character N or Y, it's considered as a string

{

vote\_flag=0;

break;//since there's no need to check the list further

}

i++;

}

if(vote\_flag)

{

red();

printf("\nSorry! You can\'t vote due to any of the following reasons-");

printf("\n\*You are not a registered voter");

printf("\n\*You have already voted\n\n");

}

fclose(file);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of voting() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 3 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*Function to be called inside main() function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void voter\_menu()

{

int option;//used for selecting menu option

blue();

printf("\n\*\*\*\*\*\*\*\*\*WELCOME to Tamil Nadu State Electoral Commission E-system\*\*\*\*\*\*\*\*\*\*");

real\_clock();//calling to display time

cyan();

printf("\t1. View Candidates\n");

printf("\t2. Register Voter\n");

printf("\t3. Vote\n");

printf("\t4. View Voting Results and Summary\n");

printf("\t5. Quit\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("\nEnter your choice: ");

scanf("%d", &option);

switch (option) {

case 1:

view\_candidates();

break;

case 2:

register\_voter();

break;

case 3:

voting();

break;

case 4:

break;

case 5:

screen\_clear();

green();

printf("\nThank you for visiting. Goodbye!\n\n");

exit(0);

break;

default:

red();

printf("\nInvalid selection, please try again!\n\n");

}

}

Progress #11

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to votecandidate.c programed by Samuela Abigail- 71762108039 \*

\* \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "votecandidate.h"

/\*\*\*\*\*global variables\*\*\*\*\*\*\*/

//for candidates

struct candidate

{

char cand\_id[45];//candidate ID

char cand\_name[45];//candidate name

char cand\_par[45];//candidate party name

int cand\_div;//candidate division number (division 1 to division 4)

int count;//number of votes of candidate

}cand;

//for voters

struct voters

{

char voter\_id[45];//voter ID

char voter\_name[45];//voter first name

char status;//Y means already voted, N means not voted

int voter\_div;//voter division number (division 1 to division 4)

int age;//age of voter should be >=18

}voter;

/\*\*\*\*\*End of global variables\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*Accessory functions\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear()

{

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function which displays real/actual time

void real\_clock()

{

time\_t t;//for showing actual time

time(&t);//I used time to make this machine look pretty, although not sure if real EVMs display time

//made this much space characters so that clock will be placed in right most corner

green();

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" %s",ctime(&t));//displays time

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of real\_clock() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*End of accessory functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*/

//displays list of all candidates

void full\_candidate\_list()

{

int i=0;

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

//inside candidate list file, if I store details having space characters inbetween, strtok is considering that also as part of string

//so strcmp will give non zero value when comparing in if loop

//so don't leave space between each detail like name and party name

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\t\t\tDIVISION\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

//https://fresh2refresh.com/c-programming/c-strings/c-strtok-function/

//https://stackoverflow.com/questions/3889992/how-does-strtok-split-the-string-into-tokens-in-c

//I tried using fscanf() with %[^|]s and all, but it didn't work

//Maybe I should have placed integers and strings alternatively in txt files and used fread properly

//anyway that didn't work correctly

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

printf("\n%s\t\t\t%s\t\t%s\t\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_div,Cand[i].Cand\_count);

}

i++;

}

printf("\n\n");

fclose(fptr);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of full\_candidate\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list division-wise

void div\_list()

{

int i=0, div\_flag=1;

char division[2]="2";//I'm initializing it as string since it is part of line in file which is a string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter division number: ");

scanf("%s",division );

purple();

printf("\nCandidates from division %s are: \n",division);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(Cand[i].Cand\_div,division)==0)

{

printf("\n%s\t\t\t%s\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_count);

div\_flag=0;

}

}

i++;

}

if(div\_flag)

{

red();

printf("\nEnter correct division number!");

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of div\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list party-wise

void par\_list()

{

int i=0, par\_flag=1;

char party[100];//for the sake of making this code work, I'm initializing it as string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter party name: ");

getchar();

scanf("%[^\n]s",party);

purple();

printf("\nCandidates from party %s are: \n",party);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tDIVISION\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(Cand[i].Cand\_par,party)==0)

{

printf("\n%s\t\t\t%s\t\t\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_div, Cand[i].Cand\_count);

par\_flag=0;

}

}

i++;

}

if(par\_flag)

{

red();

printf("\nParty not found!");

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of par\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for viewing candidates in option 1 in menu

void view\_candidates()

{

int option;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to List of Election Candidates\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\t1. View all candidates\n");

printf("\t2. View candidates by division\n");

printf("\t3. View candidates by party\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("Enter your choice: ");

scanf("%d",&option);

green();

switch (option) {

case 1:

screen\_clear();

full\_candidate\_list();

break;

case 2:

screen\_clear();

div\_list();

break;

case 3:

screen\_clear();

par\_list();

break;

default:

red();

printf("\nInvalid selection! Please try again.\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of view\_candidates() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*/

//to register as voter

void register\_voter()

{

char temp[100], line[100], str[]="@EVM#";

int i=0,j=0,len,k, dup\_flag=0, no\_space\_flag=1;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Voter Registration Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

for(k=1;k<2;k++ )//for the sake of having a loop to apply break statement.

//We have specified conditions so that for loop will be executed only once,

//otherwise it'll be infinite loop if left as for ( ; ; )

{

cyan();

printf("\*You should be at least 18 years old to register as voter.\n");

printf("\*Your division number should be from 1 to 4.\n");

printf("\*While entering name, please leave space in case of initials. Example- L Sasha\n\n\n");

yellow();

printf("\nEnter your full name: ");

getchar();//if not used means scanf won't work for strings involving space characters

scanf("%[^\n]s", voter.voter\_name);

FILE \*file=(fopen("voter\_list.txt","r"));//opening file in reading mode

if(file==NULL){

red();

printf("\nNo voters yet. Please try again later\n\n");

exit(1);

}

while ( fgets ( line, sizeof(line), file )!=NULL)//checking each line for duplicate voter name

{

char \*ptr = strstr(line, voter.voter\_name);

if (ptr != NULL)

{

dup\_flag=1;//voter name is already registered

break;//so no need to check voter list further

}

}

fclose(file);

if(dup\_flag)//breaks out of loop if flag is set to 1

{

red();

printf("Already registered as voter!\n\n");

break;

}

printf("\nEnter your age: ");

scanf("%d", &voter.age);

if(voter.age <18)

{

red();

printf("\nYou are not eligible to be vote!\n\n");

break;

}

printf("\nEnter your division number: ");

scanf("%d", &voter.voter\_div);

if(voter.voter\_div <1 || voter.voter\_div >4)

{

red();

printf("\nInvalid division number!\n\n");

break;

}

green();

printf("\n\nYou have successfully registered as a voter!");

printf("\nHere is your voter ID, PLEASE keep it carefully: ");

len=strlen(voter.voter\_name);

strcpy(temp, voter.voter\_name);//storing voter name in temp.

//If voter.voter\_name itself used means concatenated name will be saved in file in place of actual name

for(i=0,j=0 ; i<len ; i++)//for removing space characters, but won't work for names without space characters,

//and hence their voter ID's will be blank

{

if(temp[i] == ' ' && temp[i]!=NULL)

{

for(j=i ; j<len ; j++)

{

temp[j] = temp[j+1];

}

len--;

no\_space\_flag=0;//flag is set to 0 since voter name has space character

//if voter name doesn't have space character (example- Sasha),it won't go inside above loop and voter ID won't be generated

}

}

temp[j]='\0';//assigning NULL at end of string

if(no\_space\_flag)//generating voter ID for names without space character

{

strcpy(temp,voter.voter\_name);//if I remove this here despite already applying it above for loop, it's not working

strcat(temp, str);//concatenating

strcpy(voter.voter\_id,temp);//voter ID is voter name only with @EVM# in end

}

else//for voters having space character in their name

{

strcpy(voter.voter\_id,temp);//assigning voter ID

}

purple();

printf("%s\n\n", voter.voter\_id);

voter.status='N';

FILE \*fptr=(fopen("voter\_list.txt","a"));//opening file in appending mode

if(fptr==NULL){

red();

printf("\nNo voters yet. Please try again later\n\n");

exit(1);

}

//storing full name, voter ID, age, division, and status of voter in file

fprintf(fptr,"\n%s|%s|%d|%d|%c;", voter.voter\_name, voter.voter\_id, voter.age, voter.voter\_div, voter.status);

fclose(fptr);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of register\_voter() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 3 functions\*\*\*\*\*\*\*\*\*\*\*/

//registered voters will vote

void voting()

{

char buf[100], line[100], id[100], name[100];

char division[2]="2";//if I don't make division into character array (i.e) string, it won't work

int i=0,j=0, k, vote\_flag=1, voter\_is, val, voted=0;

char\* token;

struct Voter

{

char V\_id[45];//voter ID

char V\_name[45];//voter name

int V\_div;//voter division number (division 1 to division 4)

int V\_age;//voter age

int V\_status;//Y/N

}V[100];

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Election Voting Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\n\*You can vote only once");

printf("\n\*You can vote for any candidate standing in your division\n\n");

green();

printf("\nEnter your voter ID: ");

scanf("%s", id);

FILE \*file=(fopen("voter\_list.txt","r"));//opening file in reading mode

if(file==NULL){

red();

printf("\nNo voters yet. Please try again later\n\n");

exit(1);

}

for(k=1;k<2;k++ )//for the sake of having a loop to apply break statement.

//We have specified conditions so that for loop will be executed only once,

{

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), file)){

if(fscanf(file, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");//content of line till 1st '|' is encountered gets stored in token now

if(feof(file))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(V[i].V\_name,token);

token = strtok(NULL, "|");

strcpy(V[i].V\_id ,token);

token = strtok(NULL, "|");

V[i].V\_age=token;

token = strtok(NULL, "|");

V[i].V\_div = token;

token = strtok(NULL, "|");

V[i].V\_status=token;

token = strtok(NULL, "|");

}

if(strcmp(V[i].V\_id, id)==0 && strcmp(V[i].V\_status,"N")==0)

//though status is a single character N or Y, it's considered as a string

{

vote\_flag=0;

voter\_is=i;//determining which position voter is in full voter list

strcpy(division,V[i].V\_div);

yellow();

printf("\nYou are from division %s",V[i].V\_div);

}

i++;

}

if(vote\_flag)

{

red();

printf("\nSorry! You can\'t vote due to any of the following reasons-");

printf("\n\*You are not a registered voter");

printf("\n\*You have already voted\n\n");

break;

}

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

FILE \*fp;

FILE \*FP;

if ((fp = fopen("temp\_cand.txt", "w")) == NULL)

{

red();

printf("Error in opening temp\_cand.txt\n");

}

if ((FP = fopen("temp\_voter.txt", "w")) == NULL)

{

red();

printf("Error in opening temp\_voter.txt\n");

}

printf("\nCandidates from your division are: \n");

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY\t\t\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[j].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_id,token);

token = strtok(NULL, "|");

Cand[j].Cand\_div = token;

token = strtok(NULL, "|");

Cand[j].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(division,Cand[j].Cand\_div)==0)

{

printf("\n%s\t\t\t%s\t\t\t%s\t\t%s\n",Cand[j].Cand\_id,Cand[j].Cand\_name,Cand[j].Cand\_par, Cand[j].Cand\_count);

}

}

j++;

}

green();

printf("\n\nEnter name of candidate you want to vote: ");

getchar();

scanf("%[^\n]s",name);

rewind(file);

rewind(fptr);

i=0;

j=0;

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[j].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_id,token);

token = strtok(NULL, "|");

Cand[j].Cand\_div = token;

token = strtok(NULL, "|");

Cand[j].Cand\_count= token;

token = strtok(NULL, "|");

if(j==0)

{

fprintf(fp,"\n");//if first line is not empty in file, first person's details aren't being recognized

}

if(strcmp(name,Cand[j].Cand\_name)==0)

{

val=atoi(Cand[j].Cand\_count);

val++;

voted=1;

fprintf(fp,"%s|%s|%s|%s|%d;\n",Cand[j].Cand\_name,Cand[j].Cand\_par,Cand[j].Cand\_id,Cand[j].Cand\_div,val);

}

else

{

fprintf(fp,"%s|%s|%s|%s|%s;\n",Cand[j].Cand\_name,Cand[j].Cand\_par,Cand[j].Cand\_id,Cand[j].Cand\_div,Cand[j].Cand\_count);

}

}

j++;

}

while(fgets(line, sizeof(line), file)){

if(fscanf(file, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");//content of line till 1st '|' is encountered gets stored in token now

if(feof(file))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(V[i].V\_name,token);

token = strtok(NULL, "|");

strcpy(V[i].V\_id ,token);

token = strtok(NULL, "|");

V[i].V\_age=token;

token = strtok(NULL, "|");

V[i].V\_div = token;

token = strtok(NULL, "|");

V[i].V\_status=token;

token = strtok(NULL, "|");

}

if(i==0)

{

fprintf(FP,"\n");//if first line is not empty in file, first person's details aren't being recognized

}

if(i==voter\_is && voted==1)//if voted=1 condition not there, then it'll change status into Y even if voter enters wrong name

{

strcpy(V[i].V\_status,"Y");

fprintf(FP,"%s|%s|%s|%s|%s;\n",V[i].V\_name,V[i].V\_id,V[i].V\_age,V[i].V\_div,V[i].V\_status);

}

else

{

fprintf(FP,"%s|%s|%s|%s|%s;\n",V[i].V\_name,V[i].V\_id,V[i].V\_age,V[i].V\_div,V[i].V\_status);

}

i++;

}

fclose(fptr);

fclose(fp);

fclose(FP);

fclose(file);

if(voted)//if voted=1 condition not there, then it'll change status into Y even if voter enters wrong name

{

cyan();

printf("\nYou have voted successfully!");

remove("candidate\_list.txt");

remove("voter\_list.txt");

rename("temp\_cand.txt","candidate\_list.txt");

rename("temp\_voter.txt","voter\_list.txt");

}

else

{

red();

printf("\nPlease enter correct name of candidate you want to vote!");

}

printf("\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of voting() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 3 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*Function to be called inside main() function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void voter\_menu()

{

int option;//used for selecting menu option

blue();

printf("\n\*\*\*\*\*\*\*\*\*WELCOME to Tamil Nadu State Electoral Commission E-system\*\*\*\*\*\*\*\*\*\*");

real\_clock();//calling to display time

cyan();

printf("\t1. View Candidates\n");

printf("\t2. Register Voter\n");

printf("\t3. Vote\n");

printf("\t4. View Voting Results and Summary\n");

printf("\t5. Quit\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("\nEnter your choice: ");

scanf("%d", &option);

switch (option) {

case 1:

view\_candidates();

break;

case 2:

register\_voter();

break;

case 3:

voting();

break;

case 4:

break;

case 5:

screen\_clear();

green();

printf("\nThank you for visiting. Goodbye!\n\n");

exit(0);

break;

default:

red();

printf("\nInvalid selection, please try again!\n\n");

}

}

Progress #12

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to addcandidate.c programed by Merlin Might V S- 71762108027 \*

\* \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include "addcandidate.h"

#include "votecandidate.h"//to access color, screen\_clear, and real\_clock functions and struct candidate

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <ctype.h>

//inclusion of struct candidate which is already present in votecandidate.c causes error

/\*\*\*\*\*\*\*\*\*\*\*\*Global variables\*\*\*\*\*\*\*\*\*\*\*\*/

char candidateid[5];

char name[50];

char party[50];

int count, division;

char \*value;

/\*\*\*\*\*\*\*\*\*\*End of global variables\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*Menu Option 1 functions\*\*\*\*\*\*\*\*\*\*\*\*/

//for adding candidates

void add\_cand()

{

char temp[5]="", random[3],line[500];

char buffer[1024];

//if temp not initialized as empty string, @ symbol is also coming

int dup\_flag=0, nope=0, i=0, j, k, num, upper=99, lower=10;

struct Candidate

{

char id[45];//candidate ID

char name[45];//candidate name

char par[45];//candidate party name

int div;//candidate division number (division 1 to division 4)

int count;//number of votes of candidate

}C[100];

FILE\* fp = fopen("candidate\_list.txt", "r");

if (fp==NULL) {

fp = fopen("candidate\_list.txt", "w");//if candidate table doesn't exist, we have to create it

}

green();

printf("\nEnter Candidate Name: ");

getchar();

scanf("%[^\n]s", name);

for(k=1;k<2;k++)//to apply break statement

{

while (fgets(buffer,1024,fp)) {

char \*ptr = strstr(buffer, name);

if (ptr != NULL)

{

dup\_flag=1;//candidate name already there in list

break;//so no need to check candidate list further

}

}

if(dup\_flag)//breaks out of loop if flag is set to 1

{

red();

printf("\nAlready registered as candidate!\n\n");

break;

}

printf("\nEnter Candidate Party name: ");

getchar();

scanf("%[^\n]s",party);

printf("\nEnter Division number (1-4): ");

scanf("%d", &division);

if(division>4 || division<1)

{

red();

printf("\nPlease enter correct division number!\n\n");

break;

}

rewind(fp);//after checking is name is already in list,

//we have to reset position of cursor to beginning of file

//2 candidates from same party can't stand in same division

while(fgets(line, sizeof(line), fp)){

if(fscanf(fp, "%[^;]s ",buffer))

value = strtok(buffer, "|");

if(feof(fp))

break;

while (value != NULL) {

strcpy(C[i].name,value);

value = strtok(NULL, "|");

strcpy(C[i].par ,value);

value = strtok(NULL, "|");

strcpy(C[i].id,value);

value = strtok(NULL, "|");

C[i].div = value;

value = strtok(NULL, "|");

C[i].count= value;

value = strtok(NULL, "|");

if(atoi(C[i].div)==division && strcmp(party,C[i].par)==0)

{

nope=1;

break;

}

}

i++;

}

if(nope)

{

red();

printf("\nOnly 1 candidate from a party can stand in a division!\n\n");

break;

}

//for candidate ID generation

for(j=0;j<3;j++)

{

if(islower(party[j]))

{

temp[j]=party[j]-32;//to convert to uppercase

}

else

temp[j]=party[j];

}

//generating random 2-digit number for candidate ID

srand(time(0));

num = (rand() % (upper - lower + 1)) + lower;

sprintf(random,"%d",num);//copying integer to string

strcat(temp,random);//concatenating temp with random to get alphanumeric candidate ID

strcpy(candidateid,temp);

printf("\nYour Candidate ID is: ");

purple();

printf("%s",candidateid);

fclose(fp);

fp = fopen("candidate\_list.txt", "a");

// Saving data in file

count=0;//default value

fprintf(fp,"\n%s|%s|%s|%d|%d;", name, party, candidateid, division, count);

yellow();

printf("\n\nCandidate registered successfully!\n\n");

fclose(fp);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of add\_cand() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu Option 1 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*Menu Option 2 functions\*\*\*\*\*\*\*\*\*\*\*\*/

//to display full candidate list

void view\_cand()

{

int i=0;

char line[500], buffer[1024];

struct Candidate

{

char id[45];//candidate ID

char name[45];//candidate name

char par[45];//candidate party name

int div;//candidate division number (division 1 to division 4)

int count;//number of votes of candidate

}C[100];

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\t\t\tDIVISION\tVOTES\n");

FILE\* fp = fopen("candidate\_list.txt", "r");

if (fp==NULL) {

red();

printf("\nError in opening file\n");

}

yellow();

while(fgets(line, sizeof(line), fp)){

if(fscanf(fp, "%[^;]s ",buffer))

value = strtok(buffer, "|");

if(feof(fp))

break;

while (value != NULL) {

strcpy(C[i].name,value);

value = strtok(NULL, "|");

strcpy(C[i].par ,value);

value = strtok(NULL, "|");

strcpy(C[i].id,value);

value = strtok(NULL, "|");

C[i].div = value;

value = strtok(NULL, "|");

C[i].count= value;

value = strtok(NULL, "|");

printf("\n%s\t\t\t%s\t\t%s\t\t\t%s\t\t%s\n",C[i].id,C[i].name,C[i].par,C[i].div,C[i].count);

}

i++;

}

fclose(fp);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of view\_cand() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu Option 2 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*Function to be called inside main() function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void candidate\_menu()

{

int option;

blue();

printf("\n\*\*\*\*\*\*\*\*\*WELCOME to Tamil Nadu State Electoral Commission E-system\*\*\*\*\*\*\*\*\*\*");

real\_clock();//calling to display time

cyan();

printf("\t1. Add Candidate\n");

printf("\t2. View Candidates\n");

printf("\t3. Quit\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("\nEnter your choice: ");

scanf("%d", &option);

switch (option) {

case 1:

screen\_clear();

add\_cand();

break;

case 2:

screen\_clear();

view\_cand();

break;

case 3:

screen\_clear();

green();

printf("\nGood luck for the election. Goodbye!\n\n");

exit(0);

break;

default:

red();

printf("\nInvalid selection, please try again!\n\n");

}

}

Progress #13

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to votecandidate.c programed by Samuela Abigail- 71762108039 \*

\* \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "votecandidate.h"

/\*\*\*\*\*global variables\*\*\*\*\*\*\*/

//for voters

struct voters

{

char voter\_id[45];//voter ID

char voter\_name[45];//voter first name

char status;//Y means already voted, N means not voted

int voter\_div;//voter division number (division 1 to division 4)

int age;//age of voter should be >=18

}voter;

/\*\*\*\*\*End of global variables\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*Accessory functions\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear()

{

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function which displays real/actual time

void real\_clock()

{

time\_t t;//for showing actual time

time(&t);//I used time to make this machine look pretty, although not sure if real EVMs display time

//made this much space characters so that clock will be placed in right most corner

green();

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" %s",ctime(&t));//displays time

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of real\_clock() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*End of accessory functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*/

//displays list of all candidates

void full\_candidate\_list()

{

int i=0;

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

//inside candidate list file, if I store details having space characters inbetween, strtok is considering that also as part of string

//so strcmp will give non zero value when comparing in if loop

//so don't leave space between each detail like name and party name

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\t\t\tDIVISION\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

//https://fresh2refresh.com/c-programming/c-strings/c-strtok-function/

//https://stackoverflow.com/questions/3889992/how-does-strtok-split-the-string-into-tokens-in-c

//I tried using fscanf() with %[^|]s and all, but it didn't work

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

printf("\n%s\t\t\t%s\t\t%s\t\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_div,Cand[i].Cand\_count);

}

i++;

}

printf("\n\n");

fclose(fptr);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of full\_candidate\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list division-wise

void div\_list()

{

int i=0, div\_flag=1;

char division[2]="2";//I'm declaring and initializing it as string since it is part of line in file which is a string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter division number: ");

scanf("%s",division );

purple();

printf("\nCandidates from division %s are: \n",division);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(Cand[i].Cand\_div,division)==0)

{

printf("\n%s\t\t\t%s\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_count);

div\_flag=0;

}

}

i++;

}

if(div\_flag)

{

red();

printf("\nEnter correct division number!");

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of div\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for displaying candidate list party-wise

void par\_list()

{

int i=0, par\_flag=1;

char party[100];

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter party name: ");

getchar();

scanf("%[^\n]s",party);

purple();

printf("\nCandidates from party %s are: \n",party);

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tDIVISION\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(Cand[i].Cand\_par,party)==0)

{

printf("\n%s\t\t\t%s\t\t\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_div, Cand[i].Cand\_count);

par\_flag=0;

}

}

i++;

}

if(par\_flag)

{

red();

printf("\nParty not found!");

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of par\_list() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for viewing candidates in option 1 in menu

void view\_candidates()

{

int option;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to List of Election Candidates\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\t1. View all candidates\n");

printf("\t2. View candidates by division\n");

printf("\t3. View candidates by party\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("Enter your choice: ");

scanf("%d",&option);

green();

switch (option) {

case 1:

screen\_clear();

full\_candidate\_list();

break;

case 2:

screen\_clear();

div\_list();

break;

case 3:

screen\_clear();

par\_list();

break;

default:

red();

printf("\nInvalid selection! Please try again.\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of view\_candidates() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 1 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*/

//to register as voter

void register\_voter()

{

char temp[100], line[100], str[]="@EVM#";

int i=0,j=0,len,k, dup\_flag=0, no\_space\_flag=1;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Voter Registration Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

for(k=1;k<2;k++ )//for the sake of having a loop to apply break statement.

//We have specified conditions so that for loop will be executed only once,

//otherwise it'll be infinite loop if left as for ( ; ; )

{

cyan();

printf("\*You should be at least 18 years old to register as voter.\n");

printf("\*Your division number should be from 1 to 4.\n");

printf("\*While entering name, please leave space in case of initials. Example- L Sasha\n\n\n");

yellow();

printf("\nEnter your full name: ");

getchar();//if not used means scanf won't work for strings involving space characters

scanf("%[^\n]s", voter.voter\_name);

FILE \*file=fopen("voter\_list.txt","r");//opening file in reading mode

if(file==NULL){

file=fopen("voter\_list.txt","w");//since we are supposed to create a voter file if it doesn't exist

//at least that's what is written in that assignment PDF

//so I'm not printing error message nor implementing exit(1)

}

while ( fgets ( line, sizeof(line), file )!=NULL)//checking each line for duplicate voter name

{

char \*ptr = strstr(line, voter.voter\_name);

if (ptr != NULL)

{

dup\_flag=1;//voter name is already registered

break;//so no need to check voter list further

}

}

fclose(file);

if(dup\_flag)//breaks out of loop if flag is set to 1

{

red();

printf("Already registered as voter!\n\n");

break;

}

printf("\nEnter your age: ");

scanf("%d", &voter.age);

if(voter.age <18)

{

red();

printf("\nYou are not eligible to be vote!\n\n");

break;

}

printf("\nEnter your division number: ");

scanf("%d", &voter.voter\_div);

if(voter.voter\_div <1 || voter.voter\_div >4)

{

red();

printf("\nInvalid division number!\n\n");

break;

}

green();

printf("\n\nYou have successfully registered as a voter!");

printf("\nHere is your voter ID, PLEASE keep it carefully: ");

len=strlen(voter.voter\_name);

strcpy(temp, voter.voter\_name);//storing voter name in temp.

//If voter.voter\_name itself used means concatenated name will be saved in file in place of actual name

for(i=0,j=0 ; i<len ; i++)//for removing space characters, but won't work for names without space characters,

//and hence their voter ID's will be blank

{

if(temp[i] == ' ' && temp[i]!=NULL)

{

for(j=i ; j<len ; j++)

{

temp[j] = temp[j+1];

}

len--;

no\_space\_flag=0;//flag is set to 0 since voter name has space character

//if voter name doesn't have space character (example- Sasha),it won't go inside above loop and voter ID won't be generated

}

}

temp[j]='\0';//assigning NULL at end of string

if(no\_space\_flag)//generating voter ID for names without space character

{

strcpy(temp,voter.voter\_name);//if I remove this here despite already applying it above for loop, it's not working

strcat(temp, str);//concatenating

strcpy(voter.voter\_id,temp);//voter ID is voter name only with @EVM# in end

}

else//for voters having space character in their name

{

strcpy(voter.voter\_id,temp);//assigning voter ID

}

purple();

printf("%s\n\n", voter.voter\_id);

voter.status='N';

FILE \*fptr=(fopen("voter\_list.txt","a"));//opening file in appending mode

if(fptr==NULL){

red();

printf("\nNo voters yet. Please try again later\n\n");

exit(1);

}

//storing full name, voter ID, age, division, and status of voter in file

fprintf(fptr,"\n%s|%s|%d|%d|%c;", voter.voter\_name, voter.voter\_id, voter.age, voter.voter\_div, voter.status);

//if I don't use \n, it's not getting stored as separate line, but is getting attached to previous line

fclose(fptr);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of register\_voter() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 2 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 3 functions\*\*\*\*\*\*\*\*\*\*\*/

//registered voters will vote

void voting()

{

char buf[100], line[100], id[100], name[100];

char division[2]="2";//if I don't make division into character array (i.e) string, it won't work

int i=0,j=0, k, vote\_flag=1, voter\_is, val, voted=0;

char\* token;

struct Voter

{

char V\_id[45];//voter ID

char V\_name[45];//voter name

int V\_div;//voter division number (division 1 to division 4)

int V\_age;//voter age

int V\_status;//Y/N

}V[100];

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Election Voting Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\n\*You can vote only once");

printf("\n\*You can vote for any candidate standing in your division\n\n");

green();

printf("\nEnter your voter ID: ");

scanf("%s", id);

FILE \*file=(fopen("voter\_list.txt","r"));//opening file in reading mode

if(file==NULL){

red();

printf("\nNo voters yet. Please try again later\n\n");

exit(1);

}

for(k=1;k<2;k++ )//for the sake of having a loop to apply break statement.

//We have specified conditions so that for loop will be executed only once,

{

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), file)){

if(fscanf(file, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");//content of line till 1st '|' is encountered gets stored in token now

if(feof(file))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(V[i].V\_name,token);

token = strtok(NULL, "|");

strcpy(V[i].V\_id ,token);

token = strtok(NULL, "|");

V[i].V\_age=token;

token = strtok(NULL, "|");

V[i].V\_div = token;

token = strtok(NULL, "|");

V[i].V\_status=token;

token = strtok(NULL, "|");

}

if(strcmp(V[i].V\_id, id)==0 && strcmp(V[i].V\_status,"N")==0)

//though status is a single character N or Y, it's considered as a string

{

vote\_flag=0;

voter\_is=i;//determining which position this person(voter) is in full voter list

strcpy(division,V[i].V\_div);

yellow();

printf("\nYou are from division %s",V[i].V\_div);

}

i++;

}

if(vote\_flag)

{

red();

printf("\nSorry! You can\'t vote due to any of the following reasons-");

printf("\n\*You are not a registered voter");

printf("\n\*You have already voted\n\n");

break;

}

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

FILE \*fp;

FILE \*FP;

if ((fp = fopen("temp\_cand.txt", "w")) == NULL)

{

red();

printf("Error in opening temp\_cand.txt\n");

}

if ((FP = fopen("temp\_voter.txt", "w")) == NULL)

{

red();

printf("Error in opening temp\_voter.txt\n");

}

printf("\nCandidates from your division are: \n");

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY\t\t\tVOTES\n");

yellow();

// Keep printing tokens while one of the

// delimiters present in str[].

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[j].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_id,token);

token = strtok(NULL, "|");

Cand[j].Cand\_div = token;

token = strtok(NULL, "|");

Cand[j].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(division,Cand[j].Cand\_div)==0)

{

printf("\n%s\t\t\t%s\t\t\t%s\t\t%s\n",Cand[j].Cand\_id,Cand[j].Cand\_name,Cand[j].Cand\_par, Cand[j].Cand\_count);

}

}

j++;

}

green();

printf("\n\nEnter name of candidate you want to vote: ");

getchar();

scanf("%[^\n]s",name);

rewind(file);

rewind(fptr);

i=0;

j=0;

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[j].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[j].Cand\_id,token);

token = strtok(NULL, "|");

Cand[j].Cand\_div = token;

token = strtok(NULL, "|");

Cand[j].Cand\_count= token;

token = strtok(NULL, "|");

if(j==0)

{

fprintf(fp,"\n");//if first line is not empty in file, first person's details aren't being recognized

}

if(strcmp(name,Cand[j].Cand\_name)==0 && strcmp(division,Cand[j].Cand\_div )==0)

//if voting is done without checking if voter and candidate are from same division, it won't be foolproof

{

val=atoi(Cand[j].Cand\_count);

val++;

voted=1;

fprintf(fp,"%s|%s|%s|%s|%d;\n",Cand[j].Cand\_name,Cand[j].Cand\_par,Cand[j].Cand\_id,Cand[j].Cand\_div,val);

}

else

{

fprintf(fp,"%s|%s|%s|%s|%s;\n",Cand[j].Cand\_name,Cand[j].Cand\_par,Cand[j].Cand\_id,Cand[j].Cand\_div,Cand[j].Cand\_count);

}

}

j++;

}

while(fgets(line, sizeof(line), file)){

if(fscanf(file, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");//content of line till 1st '|' is encountered gets stored in token now

if(feof(file))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(V[i].V\_name,token);

token = strtok(NULL, "|");

strcpy(V[i].V\_id ,token);

token = strtok(NULL, "|");

V[i].V\_age=token;

token = strtok(NULL, "|");

V[i].V\_div = token;

token = strtok(NULL, "|");

V[i].V\_status=token;

token = strtok(NULL, "|");

}

if(i==0)

{

fprintf(FP,"\n");//if first line is not empty in file, first person's details aren't being recognized

}

if(i==voter\_is && voted==1)//if voted=1 condition not there, then it'll change status into Y even if voter enters wrong name

{

strcpy(V[i].V\_status,"Y");

fprintf(FP,"%s|%s|%s|%s|%s;\n",V[i].V\_name,V[i].V\_id,V[i].V\_age,V[i].V\_div,V[i].V\_status);

}

else

{

fprintf(FP,"%s|%s|%s|%s|%s;\n",V[i].V\_name,V[i].V\_id,V[i].V\_age,V[i].V\_div,V[i].V\_status);

}

i++;

}

fclose(fptr);

fclose(fp);

fclose(FP);

fclose(file);

if(voted)//if voted=1 condition not there, then it'll change status into Y even if voter enters wrong name

{

cyan();

printf("\nYou have voted successfully!");

remove("candidate\_list.txt");

remove("voter\_list.txt");

rename("temp\_cand.txt","candidate\_list.txt");

rename("temp\_voter.txt","voter\_list.txt");

}

else

{

red();

printf("\nPlease enter correct name of candidate you want to vote!");

}

printf("\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of voting() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 3 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*Menu option 4 functions\*\*\*\*\*\*\*\*\*\*\*/

//voting results and summary

void voting\_result()

{

int option;

screen\_clear();

blue();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME to Voting Results Portal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

real\_clock();

cyan();

printf("\t1. View results of all candidates\n");

printf("\t2. View results of candidates in a specific division\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("Enter your choice: ");

scanf("%d",&option);

green();

switch (option) {

case 1:

screen\_clear();

full\_result();

break;

case 2:

screen\_clear();

div\_result();

break;

default:

red();

printf("\nInvalid selection! Please try again.\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of voting\_result() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for viewing results of all candidates

void full\_result()

{

/\*int i=0;

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

}Cand[100];

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

purple();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\t\t\tDIVISION\tVOTES\n");

yellow();

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

printf("\n%s\t\t\t%s\t\t%s\t\t\t%s\t\t%s\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_div,Cand[i].Cand\_count);

}

i++;

}

printf("\n\n");

fclose(fptr);\*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of full\_result() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//for viewing results of candidates in specific division

void div\_result()

{

int i=0, j=0, k, total\_cand\_div=0, div\_flag=1, total\_votes=0;

int highest=0, lowest=1;//for finding candidate with max and min votes respectively

//if I set lowest=0, then for lowest<C[i].Cand\_count condition no one will get stored

char division[2]="2";//I'm declaring and initializing it as string since it is part of line in file which is a string

char buf[500], line[500];

char\* token;

struct Candidate

{

char Cand\_id[45];//candidate ID

char Cand\_name[45];//candidate name

char Cand\_par[45];//candidate party name

int Cand\_div;//candidate division number (division 1 to division 4)

int Cand\_count;//number of votes of candidate

float Cand\_per;//percentage of votes

}Cand[100];

//for candidate with highest votes

struct winner

{

char id[45];//candidate ID

char name[45];//candidate name

char par[45];//candidate party name

int count;//number of votes of candidate

float percent;//percentage of votes received

}win;

//for candidate with lowest votes

struct last

{

char id[45];//candidate ID

char name[45];//candidate name

char par[45];//candidate party name

int count;//number of votes of candidate

float percent;//percentage of votes received

}loss;

FILE \*fptr;

if ((fptr = fopen("candidate\_list.txt", "r")) == NULL)

{

red();

printf("Candidate list is empty.\n");

}

green();

printf("Enter division number: ");

scanf("%s",division );

for(k=1;k<2;k++){

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(Cand[i].Cand\_div,division)==0)

{

total\_cand\_div++;//finding total number of candidates in that division to calculate % of votes

total\_votes+=atoi(Cand[i].Cand\_count);//finding total votes in that division

div\_flag=0;//means entered division number is valid

if(highest<atoi(Cand[i].Cand\_count))

{

strcpy(win.name, Cand[i].Cand\_name);

strcpy(win.id, Cand[i].Cand\_id);

strcpy(win.par, Cand[i].Cand\_par);

win.count=atoi(Cand[i].Cand\_count);

}

if(lowest>atoi(Cand[i].Cand\_count))

{

strcpy(loss.name, Cand[i].Cand\_name);

strcpy(loss.id, Cand[i].Cand\_id);

strcpy(loss.par, Cand[i].Cand\_par);

loss.count=atoi(Cand[i].Cand\_count);

}

}

}

i++;

}

if(div\_flag)

{

red();

printf("\nEnter correct division number!");

break;

}

purple();

printf("\nVoting results and summary from division %s: \n",division);

green();

printf("\nTotal candidates contesting in this division: %d",total\_cand\_div);

printf("\nTotal votes received in this division: %d\n",total\_votes);

cyan();

printf("\nCandidate who secured highest votes: ");

yellow();

printf("\nCandidate Name: %s",win.name);

printf("\nCandidate ID: %s",win.id);

printf("\nParty: %s",win.par);

printf("\nVotes received: %d",win.count);

win.percent=(win.count\*100)/total\_cand\_div;

printf("\nPercentage of votes: %f\n",win.percent);

cyan();

printf("\nCandidate who secured least votes: ");

yellow();

printf("\nCandidate Name: %s",loss.name);

printf("\nCandidate ID: %s",loss.id);

printf("\nParty: %s",loss.par);

printf("\nVotes received: %d",loss.count);

loss.percent=(loss.count\*100)/total\_cand\_div;

printf("\nPercentage of votes: %f\n",loss.percent);

purple();

printf("\nFull voting results of division %s: \n",division);

blue();

printf("\nCANDIDATE ID\t\t\tNAME\t\t\t\tPARTY NAME\tVOTES\tPERCENTAGE OF VOTES\n");

rewind(fptr);

yellow();

while(fgets(line, sizeof(line), fptr)){

if(fscanf(fptr, "%[^;]s ",buf))//scans till ; is encountered and stores that string in buf

token = strtok(buf, "|");

if(feof(fptr))//if I let this while loop terminate on it's own, full program gets terminated abruptly without executing other codes

break;

while (token != NULL) {

strcpy(Cand[i].Cand\_name,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_par ,token);

token = strtok(NULL, "|");

strcpy(Cand[i].Cand\_id,token);

token = strtok(NULL, "|");

Cand[i].Cand\_div = token;

token = strtok(NULL, "|");

Cand[i].Cand\_count= token;

token = strtok(NULL, "|");

if(strcmp(Cand[i].Cand\_div,division)==0)

{

Cand[i].Cand\_per=(atoi(Cand[i].Cand\_count)\*100)/total\_cand\_div;

printf("\n%s\t\t\t%s\t\t%s\t\t%s\t%f\n",Cand[i].Cand\_id,Cand[i].Cand\_name,Cand[i].Cand\_par,Cand[i].Cand\_count,Cand[i].Cand\_per);

}

}

i++;

}

}

fclose(fptr);

printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of div\_result() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*End of Menu option 4 functions\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*Function to be called inside main() function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void voter\_menu()

{

int option;//used for selecting menu option

blue();

printf("\n\*\*\*\*\*\*\*\*\*WELCOME to Tamil Nadu State Electoral Commission E-system\*\*\*\*\*\*\*\*\*\*");

real\_clock();//calling to display time

cyan();

printf("\t1. View Candidates\n");

printf("\t2. Register Voter\n");

printf("\t3. Vote\n");

printf("\t4. View Voting Results and Summary\n");

printf("\t5. Quit\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLEASE select a choice\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

yellow();

printf("\nEnter your choice: ");

scanf("%d", &option);

switch (option) {

case 1:

view\_candidates();

break;

case 2:

register\_voter();

break;

case 3:

voting();

break;

case 4:

voting\_result();

break;

case 5:

screen\_clear();

green();

printf("\nThank you for visiting. Goodbye!\n\n");

exit(0);

break;

default:

red();

printf("\nInvalid selection, please try again!\n\n");

}

}