-_
/*
Requirements
1. Define Data Types
Candidate Structure:

Use a structure Candidate to represent details of candidates contesting in elections with the following fields:
candidateID (integer): Unique identifier for the candidate.
candidateName (string): Name of the candidate.
partyName (string): Party represented by the candidate.
votesReceived (integer): Number of votes received by the candidate.
constituency (string): Name of the constituency.
Voter Structure:

Use a structure Voter to represent voter details:
voterID (integer): Unique identifier for the voter.
voterName (string): Name of the voter.
age (integer): Age of the voter.
hasVoted (boolean): Indicates if the voter has voted or not.

Union for Voter/Candidate Details:

Use a union ElectionEntityDetails to store either voter details or candidate details depending on the context:
Candidate candidateData: Details of a candidate.
Voter voterData: Details of a voter.

2. Features
Dynamic Memory Allocation:

Dynamically allocate memory for:
An array of Candidate structures based on the number of candidates.
An array of Voter structures based on the number of voters.
Input and Output:
Input details for all candidates and voters.
Input voting data, where voters vote for candidates by their candidateID.
Display:
Display all candidates, their parties, and the total votes they received.
Display voter details, including whether they have voted or not.
Search:
Search for a candidate by candidateID or a voter by voterID.
Election Result:
Calculate and display the winner based on the highest votes received.
Handle ties by listing all candidates with the highest votes.
Sorting:
Sort candidates by votes received in descending order.
Sort voters by their voterID in ascending order.

3. Typedef
Use typedef to simplify declarations of Candidate, Voter, and ElectionEntityDetails.

Program Requirements
1. Menu Options
Input Candidate Details.
Input Voter Details.
Record Voting.

Display All Candidate Details.
Display All Voter Details.
Search Candidate by ID.
Search Voter by ID.
Display Election Result.
Exit.

*/

-_

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>

typedef struct candidate{
   int candidateID;
   char candidateName[20];
   char partyName[20];
   int votesReceived;
   char constituency[20];
}CANDIDATE;

typedef struct voter{
   int voterID;
   char voterName[20];
   int voterAge;
   bool hasVoted;
}VOTER;

typedef union voter_candidate_details{
   CANDIDATE C;
   VOTER V;
}V_C_Details;

void addCANDIDATE(CANDIDATE *,int);
void addVOTER(VOTER *,int);
void displayCANDIDATE(CANDIDATE *,int);
void displayVOTER(VOTER *,int);
void searchCANDIDATE(CANDIDATE *,int,int);
void recordVOTING(CANDIDATE*,VOTER*,int,int);
void searchByCandidateID(CANDIDATE *,int,int);
void searchByVoterID(VOTER *,int,int);
void electionResult(CANDIDATE*,VOTER*,int,int);

int main(){
   int op,noC,noV;
   printf("menu:\n");
   printf("1-Input Candidate Details.\n2-Input Voter Details.\n3-Record Voting.\n4Display All Candidate Details.\n5Display All Voter Details.\n6Search Candidate by ID.\n7Search Voter by ID.\n8Display Election Result.\n9Exit.");
```

```
  while(1){
    printf("\nenter operation:");
    scanf("%d",&op);
    switch(op){
      case 1:
        printf("enter number of cadidates:");
        scanf("%d",&noC);
        CANDIDATE *Candidate_Details=(CANDIDATE*)malloc(noC*sizeof(CANDIDATE));
        addCANDIDATE(Candidate_Details,noC);
        break;
      case 2:
        printf("enter number of voters:");
        scanf("%d",&noV);
        VOTER *Voter_Details=(VOTER*)malloc(noV*sizeof(VOTER));
        addVOTER(Voter_Details ,noV);
        break;
      case 3:
        recordVOTING(Candidate_Details,Voter_Details,noC,noV);
        break;
      case 4:
        displayCANDIDATE(Candidate_Details,noC);
        break;
      case 5:
        displayVOTER(Voter_Details,noV);
        break;
      case 6:
        int Candidate_ID;
        printf("enter the candidate id to be searched:");
        scanf("%d",&Candidate_ID);
        searchByCandidateID(Candidate_Details,noC,Candidate_ID);
        break;
      case 7:
        int Voter_ID;
        printf("enter the voter id to be searched:");
        scanf("%d",&Voter_ID);
        searchByVoterID(Voter_Details,noV,Voter_ID);
        break;
      case 8:
        electionResult(Candidate_Details,Voter_Details,noC,noV);
        break;
      case 9:
        free(Candidate_Details);
        free(Voter_Details);
        printf("exiting system\n");
        return 0;

      default:
        printf("invalid operation\n");

    }
  }

  return 0;
}
void addCANDIDATE(CANDIDATE *Candidate_Details,int noC){
```

```
  printf("enter Candidate Details\n");
  for(int i=0;i<noC;i++){
    printf("enter %d candidate\n",i+1);
    printf("Name:");
    scanf(" %[^\n]",(Candidate_Details+i)->candidateName);
    printf("candidate ID:");
    scanf("%d",&(Candidate_Details+i)->candidateID);
    printf("Party Name:");
    scanf(" %[^\n]",(Candidate_Details+i)->partyName);
    printf("constituency:");
    scanf(" %[^\n]",(Candidate_Details+i)->constituency);
    (Candidate_Details+i)->votesReceived=0;

  }
}
void addVOTER(VOTER *Voter_Details,int noV){
  printf("enter Voter Details\n");
  for(int i=0;i<noV;i++){
    printf("enter %d voter\n",i+1);
    printf("Name:");
    scanf(" %[^\n]",(Voter_Details+i)->voterName);
    printf("voter ID:");
    scanf("%d",&(Voter_Details+i)->voterID);

    printf("voter age:");
    scanf("%d",&(Voter_Details+i)->voterAge);
    if((Voter_Details+i)->voterAge>=18){
      printf("voter added successfully\n");
    }
    else{
      printf("cannot add voter\n");
      printf("underage\n");
      i--;
    }
    (Voter_Details+i)->hasVoted=false;

  }
}

void displayCANDIDATE(CANDIDATE *Candidate_Details,int noC){
  printf("all candidate details\n");
  printf("name|\t\t|ID|\t\t|Party|\t\t|constituency|\t\t|votes Received\n");
  for(int i=0;i<noC;i++){
    printf("%s|\t\t|%d|\t\t|%s|\t\t|%s|\t\t|%d\n",(Candidate_Details+i)->candidateName,(Candidate_Details+i)-
>candidateID,(Candidate_Details+i)->partyName,
          (Candidate_Details+i)->constituency,(Candidate_Details+i)->votesReceived);
  }
}
void displayVOTER(VOTER *Voter_Details,int noV){
  printf("all voter details\n");
  printf("name|\t\t|ID|\t|\t|age\n");
  for(int i=0;i<noV;i++){
    printf("%s|\t|%d|\t|%d\n",(Voter_Details+i)->voterName,(Voter_Details+i)->voterID,(Voter_Details+i)->voterAge);
  }
}
```

```
void recordVOTING(CANDIDATE *Candidate_Details, VOTER *Voter_Details, int noC, int noV) {
  int voters[50] = {0};
  int voterCount = 0;

  printf("Candidates:\n");
  for (int i = 0; i < noC; i++) {
    printf("ID: %d, Name: %s, Party: %s\n",
        (Candidate_Details + i)->candidateID,
        (Candidate_Details + i)->candidateName,
        (Candidate_Details + i)->partyName);
  }

  for (int i = 0; i < noV; i++) {
    printf("\nEnter Voter ID: ");
    int voterID;
    scanf("%d", &voterID);

    int voterIndex = -1;
    for (int j = 0; j < noV; j++) {
      if (Voter_Details[j].voterID == voterID && !Voter_Details[j].hasVoted) {
        voterIndex = j;
        break;
      }
    }

    if (voterIndex == -1) {
      printf("Invalid or already used Voter ID.\n");
      continue;
    }

    printf("Enter Candidate ID to vote for: ");
    int candidateID;
    scanf("%d", &candidateID);

    int candidateIndex = -1;
    for (int j = 0; j < noC; j++) {
      if ((Candidate_Details+j)->candidateID == candidateID) {
        candidateIndex = j;
        break;
      }
    }

    while(candidateIndex == -1) {
        printf("Invalid Candidate ID.\n");
        printf("re_enter\n");
        scanf("%d", &candidateID);
        for (int j = 0; j < noC; j++) {
          if ((Candidate_Details+j)->candidateID == candidateID) {
            candidateIndex = j;
            break;
          }
        }

    }
```

```
    (Candidate_Details+candidateIndex)->votesReceived++;
    (Voter_Details+voterIndex)->hasVoted = true;
    printf("Vote recorded.\n");
  }
}

void searchByCandidateID(CANDIDATE *Candidate_Details,int noC,int ID){
  int found=0;
  int i=0;
  for(int i=0;i<noC;i++){
    if(ID==(Candidate_Details+i)->candidateID){
      found=1;
      break;
    }
  }
  if(found==1){
    printf("CANDIDATE found\n");
    printf("ID: %d, Name: %s, Party: %s\n",
        (Candidate_Details + i)->candidateID,
        (Candidate_Details + i)->candidateName,
        (Candidate_Details + i)->partyName);

  }
  else{
    printf("invalid id\n");
  }
}
void searchByVoterID(VOTER *Voter_Details,int noV,int ID){
  int found=0;
  int i=0;
  for(int i=0;i<noV;i++){
    if(ID==(Voter_Details+i)->voterID){
      found=1;
      break;
    }
  }
  if(found==1){
    printf("Voter found\n");
    printf("ID: %d, Name: %s, Age: %d\n",
        (Voter_Details + i)->voterID,
        (Voter_Details + i)->voterName,
        (Voter_Details + i)->voterAge);

  }
  else{
    printf("invalid id\n");
  }
}
void electionResult(CANDIDATE *Candidate_Details,VOTER *Voter_Details,int noC,int noV){
  int max=Candidate_Details->votesReceived;
  int max_count=0;
  for(int i=0;i<noC-1;i++){
    if(max<(Candidate_Details+i+1)->votesReceived){
      max=(Candidate_Details+i+1)->votesReceived;
      max_count=i;
```

```
    }
  }
  printf("Election result\n");
  displayCANDIDATE(Candidate_Details,noC);
  printf("%s elected \n",(Candidate_Details + max_count)->candidateName);
}
```