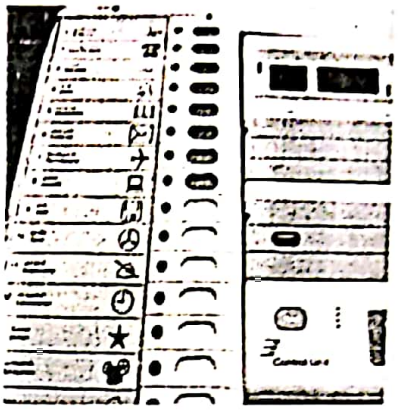
Objective:

- It will help in understanding the use of struct with pointer as data member and array of struct objects.

Challenge: ***Voting Machine***

Our objective is to make an application to conduct election and compute results for candidates contesting within some particular constituency.



The struct Candidate is used to store the information about a candidate who is contesting in election.
The struct VotingMachine keeps records of all the candidates contesting in election and is used to cast votes for the candidates in a constituency.

```
struct Candidate
{
    char name[100];           //name of candidate
    char electionSymbol[30]; //election symbol of candidate
    int votes;                //number of votes casted to candidate
};
```

Operations for the Candidate struct

1. void inputCandidate (Candidate *);
   input candidate name and election symbol. Obviously, Votes will be initialized to 0 instead of taking input ☺.

2. void castVote(Candidate &);
   increment by 1 in the vote count of the received candidate

```
struct Time
{
    int hours;
    int minutes;
    int seconds;
};
```

```
struct VotingMachine
{
    char constituencyName[20];   // like NA-105 or PP-404
    Date electionDate;           // stores election date. Same struct as we used previously.
    Time startTime;              // stores election start time.
    Time endTime;                // stores election end time.
                                 // candidates can't be added in machine on the electionDate.
    Candidate * candidateList;   // points to an array of candidate objects contesting in
                                 // elections
    int numOfCandidates;         // number of candidate objects stored in machine
    int capacity;                // capacity of machine to store candidates in machine. i.e., it
                                 // stores the size of array pointed by candidateList.
};
```

Stores information related to Time in 24 hours format.
So, hours will have value from 0 to 23
minutes and seconds will have value from 0 to 59

Operations for the VotingMachine struct

1. *void initializeVotingMachine* (VotingMachine & vm, const char * constName, Date d, Time sTime, Time eTime, int cap);
   It initializes the voting machine with given data.
   constName is initialized to constituencyName of voting machine.
   d with electionDate of voting machine.
   sTime with startTime of voting machine.
   eTime with endTime of voting machine.
   cap with capacity of voting machine.
   numOfCandidates with 0.
   candidateList points to an array of Candidate objects of size capacity.

2. bool addCandidate (VotingMachine & vm, const char * candName, const char * elecSymb);
   Add the candidate object with given candidate name and election symbol in the candidateList.
   Make sure that a candidate with duplicate election symbol must not be added.

It returns true if candidate is added successfully otherwise false.

3. bool castVote(VotingMachine & vm, const char * elecSymbol);
   It searches the candidate in Voting Machine with given election symbol and increments vote count by calling castVote function on the particular candidate object. You will not directly increment the vote count rather your code must call/use castVote function defined for candidate for the said purpose.
   Note: This function will not cast vote if the system date is not equal to the date stored in voting machine. Same goes for the time, the time of cast vote must be within the range of voting machine start time and end time. How to get the system time and date is elaborated at the end.
   It returns true if vote is casted successfully otherwise false.

4. int candidateReport(VotingMachine & vm, const char * elecSymbol);
   It returns the number of votes casted so far to the candidate whose election symbol is received. Return -1 if elecSymbol is not found.

5. Candidate * electionResult (VotingMachine & vm);
   It returns an array of candidates having 3 objects with winner at index 0, runner-up at index 1 and 3rd position holder at index 2.

6. void freeVotingMachine (VotingMachine & vm);
   It deallocates the memory resources captured by voting machine.

Sample Run

```
int main()
{
    VotingMachine vMachine;
    Date d = {23,10,2021};
    Time st = {8,0,0};
    Time et = {18,0,0};
    initializeVotingMachine(vMachine, "PP-404", d, st, et,
10);

     addCandidate(vMachine, "Aslam", "Racket");
     addCandidate(vMachine, "Naeem", "Kulhara");
     addCandidate(vMachine, "Ayesha", "abc");
     addCandidate(vMachine, "Rabia", "zzz");
     addCandidate(vMachine, "Aftab", "Hockey");
     addCandidate(vMachine, "Manan", "TV");

      /* If you notice we have to add candidates in machine at
      least 1 day before the election date. But this way, in lab,
      we shall not be able to test the code by waiting for next
      day for voting.
      So, after adding candidates in machine, we change the
      election date as follows: */

       d.day = 22; //our actual date of election
       vMachine.electionDate = d;

      //now rest of code may work successfully.

      castVote(vMachine, "zzz");
      castVote(vMachine, "Kulhara");
      castVote(vMachine, "zzz");
      castVote(vMachine, "zzz");
      castVote(vMachine, "Hockey");
      castVote(vMachine, "Hockey");
      castVote(vMachine, "zzz");
      castVote(vMachine, "Hockey");
      castVote(vMachine, "Hockey");

    cout<<"Vote count so far for Hockey:
"<<candidateReport(vMachine, "Hockey");

    castVote(vMachine, "Hockey");
```

Console Output

```
Vote count so far for Hockey: 4

*******Election Results*******
1st Position: Hockey : 6
2nd Position: zzz : 4
3rd Position: Racket : 1
```

```
    castVote(vMachine, "Hockey");
    castVote(vMachine, "TV");
    castVote(vMachine, "abc");
    castVote(vMachine, "Racket");

    Candidate * list = electionResult(vMachine);
    cout<<"\n\n*******Election Results*******\n";
    cout<<"1st Position: "<<list[0].electionSymbol<<" :
"<<list[0].votes<<'\n';
    cout<<"2nd Position: "<<list[1].electionSymbol<<" :
"<<list[1].votes<<'\n';
    cout<<"3rd Position: "<<list[2].electionSymbol<<" :
"<<list[2].votes<<'\n';

    delete [] list;
    freeVotingMachine(vMachine);

    cout<<"\n\n";
    return 0;
}
```

How to get current/system date/time?

CTime library will help us in this regard, which you may explore in detail at home as per your interest but for now I am pasting code which may give you required stuff for this lab at least.

```
time_t t = time(NULL);
tm curTime = * localtime(&t);
cout << "Current Date: " << curTime.tm_mday << '-' << curTime.tm_mon + 1 << '-' <<
        curTime.tm_year + 1900 << "\n";
cout << "Current Time: " << curTime.tm_hour << ':' << curTime.tm_min << ':' << curTime.tm_sec
        << "\n";
```

A short explanation of the stuff used above:

- `time(NULL)` returns the time since 00:00:00 UTC, January 1, 1970 in seconds.
- `time_t` is an alias of integral data type capable of holding value returned by `time(NULL)`
- `localtime` function converts the received `time_t` object into calendar time. It actually returns an object of type tm struct whose attributes are as follows:

Member	Type	Meaning	Range
tm_sec	int	seconds after the minute	0-61*
tm_min	int	minutes after the hour	0-59
tm_hour	int	hours since midnight	0-23
tm_mday	int	day of the month	1-31
tm_mon	int	months since January	0-11
tm_year	int	years since 1900	
tm_wday	int	days since Sunday	0-6
tm_yday	int	days since January 1	0-365
tm_isdst	int	Daylight Saving Time flag	

"In looking for people to hire, you look for three qualities: integrity, intelligence, and energy. And, if they don't have the first, the other two will kill you."

--Warren Buffett--