This is CS50

CS50's Introduction to Computer Science

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/) (https://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Plurality

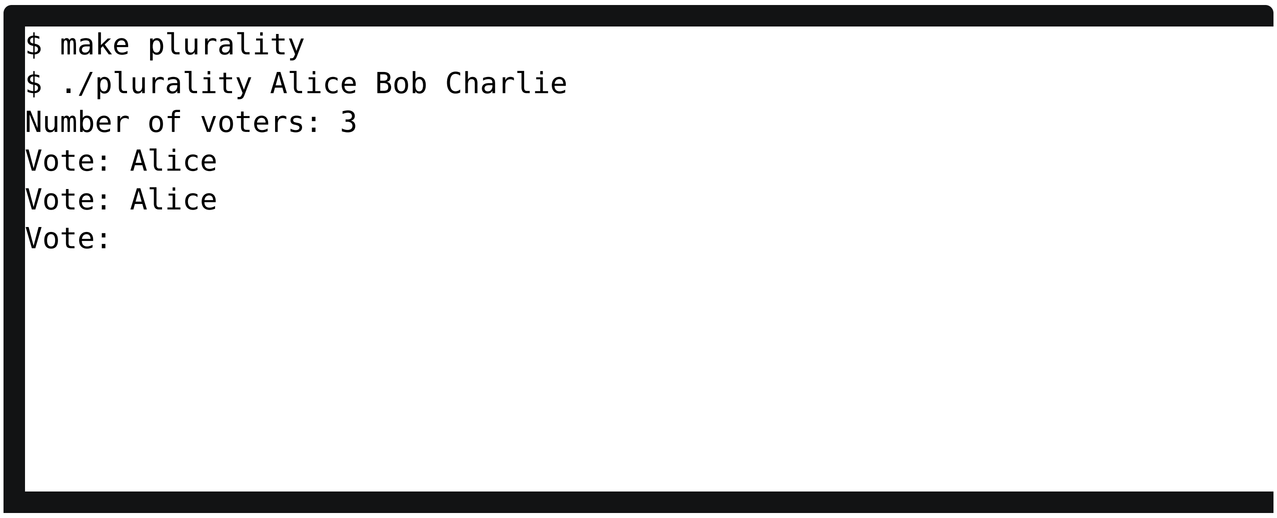
Problem to Solve

Elections come in all shapes and sizes. In the UK, the Prime Minister (https://www.parliament.uk/education/about-your-parliament/general-elections/) is officially appointed by the monarch, who generally chooses the leader of the political party that wins the most seats in the House of Commons. The United States uses a multi-step Electoral College (https://www.archives.gov/federal-register/electoral-college/about.html) process where citizens vote on how each state should allocate Electors who then elect the President.

Perhaps the simplest way to hold an election, though, is via a method commonly known as the "plurality vote" (also known as "first-past-the-post" or "winner take all"). In the plurality vote, every voter gets to vote for one candidate. At the end of the election, whichever candidate has the greatest number of votes is declared the winner of the election.

For this problem, you'll implement a program that runs a plurality election, per the below.

Demo



```
$ make plurality
$ ./plurality Alice Bob Charlie
Number of voters: 3
Vote: Alice
Vote: Alice
Vote:
```

Recorded with asciinema

Distribution Code

For this problem, you'll extend the functionality of "distribution code" provided to you by CS50's staff.

▼ **Download the distribution code**

Log into cs50.dev (https://cs50.dev/), click on your terminal window, and execute `cd` by itself. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
wget https://cdn.cs50.net/2023/fall/psets/3/plurality.zip
```

in order to download a ZIP called `plurality.zip` into your codespace.

Then execute

```
unzip plurality.zip
```

to create a folder called `plurality`. You no longer need the ZIP file, so you can execute

```
rm plurality.zip
```

and respond with "y" followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd plurality
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
plurality/ $
```

If all was successful, you should execute

```
ls
```

and see a file named `plurality.c`. Executing `code plurality.c` should open the file where you will type your code for this problem set. If not, retrace your steps and see if you can determine where you went wrong!

Hints

▾ Understand the code in `plurality.c`

Whenever you'll extend the functionality of existing code, it's best to be sure you first understand it in its present state.

Look first at the top of the file. The line `#define MAX 9` is some syntax used here to mean that `MAX` is a constant (equal to `9`) that can be used throughout the program. Here, it represents the maximum number of candidates an election can have.

```
// Max number of candidates
#define MAX 9
```

Notice that `plurality.c` then uses this constant to define a global array—that is, an array that any function can access.

```
// Array of candidates
candidate candidates[MAX];
```

But what, in this case, is a `candidate`? In `plurality.c`, a `candidate` is a `struct`. Each `candidate` has two fields: a `string` called `name` representing the candidate's name, and an `int` called `votes` representing the number of votes the candidate has.

```
// Candidates have name and vote count
typedef struct
{
    string name;
    int votes;
}
candidate;
```

Now, take a look at the `main` function itself. See if you can find where the program sets a global variable `candidate_count` representing the number of candidates in the election.

```
// Number of candidates
int candidate_count;
```

What about where it copies command-line arguments into the array `candidates`?

```
// Populate array of candidates
candidate_count = argc - 1;
if (candidate_count > MAX)
{
    printf("Maximum number of candidates is %i\n", MAX);
    return 2;
}
for (int i = 0; i < candidate_count; i++)
{
    candidates[i].name = argv[i + 1];
    candidates[i].votes = 0;
}
```

And where it asks the user to type in the number of voters?

```
int voter_count = get_int("Number of voters: ");
```

Then, the program lets every voter type in a vote, calling the `vote` function on each candidate voted for. Finally, `main` makes a call to the `print_winner` function to print out the winner (or winners) of the election. We'll leave it to you to identify the code that implements this functionality.

If you look further down in the file, though, you'll notice that the `vote` and `print_winner` functions have been left blank.

```
// Update vote totals given a new vote
bool vote(string name)
{
    // TODO
    return false;
}

// Print the winner (or winners) of the election
void print_winner(void)
{
    // TODO
    return;
}
```

This part is up to you to complete! **You should not modify anything else in `plurality.c` other than the implementations of the `vote` and `print_winner` functions (and the inclusion of additional header files, if you'd like).**

▼ Complete the `vote` function

Next, complete the `vote` function.

- Consider that `vote`'s signature, `bool vote(string name)`, shows it takes a single argument, a `string` called `name`, representing the name of the candidate who was voted for.
- `vote` should return a `bool`, where `true` indicates a vote was successfully cast and `false` indicates it was not.

One way to approach this problem is to do the following:

1. Iterate over each candidate
    1. Check if candidate's name matches the input, `name`
        1. If yes, increment that candidate's votes and return `true`
        2. If no, continue checking
2. If no matches after checking each candidate, return `false`

Let's write some pseudocode to remind you to do just that:

```
// Update vote totals given a new vote
bool vote(string name)
{
    // Iterate over each candidate
        // Check if candidate's name matches given name
            // If yes, increment candidate's votes and return true

    // If no match, return false
}
```

We'll leave the implementation in code, though, up to you!

▼ Complete the `print_winner` function

Finally, complete the `print_winner` function.

- The function should print out the name of the candidate who received the most votes in the election, and then print a newline.
- The election could end in a tie if multiple candidates each have the maximum number of votes. In that case, you should output the names of each of the winning candidates, each on a separate line.

You might think a sorting algorithm would best solve this problem: imagine sorting candidates by their vote totals and printing the top candidate (or candidates). Recall, though, that sorting can be expensive: even Merge Sort, one of the fastest sorting algorithms, runs in $O(N\ log(N))$.

Consider that you need only two pieces of information to solve this problem:

1. The maximum number of votes
2. The candidate (or candidates) with that number of votes

As such, a good solution might require only two searches. Write some pseudocode to remind yourself to do just that:

```
// Print the winner (or winners) of the election
void print_winner(void)
{
    // Find the maximum number of votes

    // Print the candidate (or candidates) with maximum votes

}
```

We'll leave the code, though, up to you!

Walkthrough

How to Test

Be sure to test your code to make sure it handles…

- An election with any number of candidate (up to the `MAX` of `9`)
- Voting for a candidate by name
- Invalid votes for candidates who are not on the ballot
- Printing the winner of the election if there is only one
- Printing the winner of the election if there are multiple winners

Correctness

```
check50 cs50/problems/2024/x/plurality
```

Style

```
style50 plurality.c
```

How to Submit

```
submit50 cs50/problems/2024/x/plurality
```