Isso é CS50

Introdução do CS50 à Ciência da Computação

OpenCourseWare

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

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.clubhouse.com/@davidjmalan) (https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/) (https://orcid.org/0000-0001-5338-2522) (https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan) (https://www.tiktok.com/@davidjmalan) (https://davidjmalan.t.me/) (https://twitter.com/davidjmalan)

Escoamento

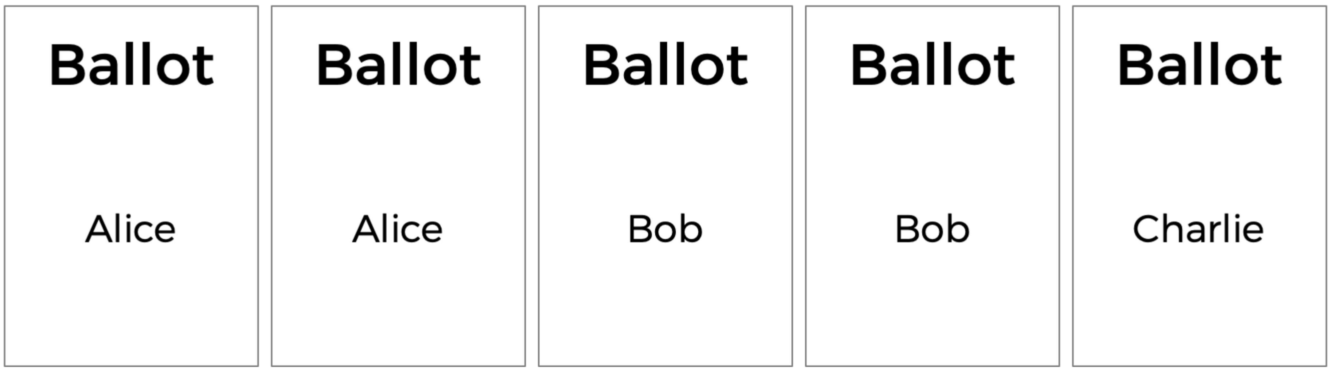
Para este programa, você implementará um programa que executa uma eleição de segundo turno, conforme abaixo.

```
./runoff Alice Bob Charlie
Number of voters: 5
Rank 1: Alice
Rank 2: Bob
Rank 3: Charlie

Rank 1: Alice
Rank 2: Charlie
Rank 3: Bob

Rank 1: Bob
Rank 2: Charlie
Rank 3: Alice

Rank 1: Bob
Rank 2: Alice
Rank 3: Charlie

Rank 1: Charlie
Rank 2: Alice
Rank 3: Bob

Alice
```

Fundo

Você já conhece as eleições de pluralidade, que seguem um algoritmo muito simples para determinar o vencedor de uma eleição: cada eleitor recebe um voto, e o candidato com mais votos vence.

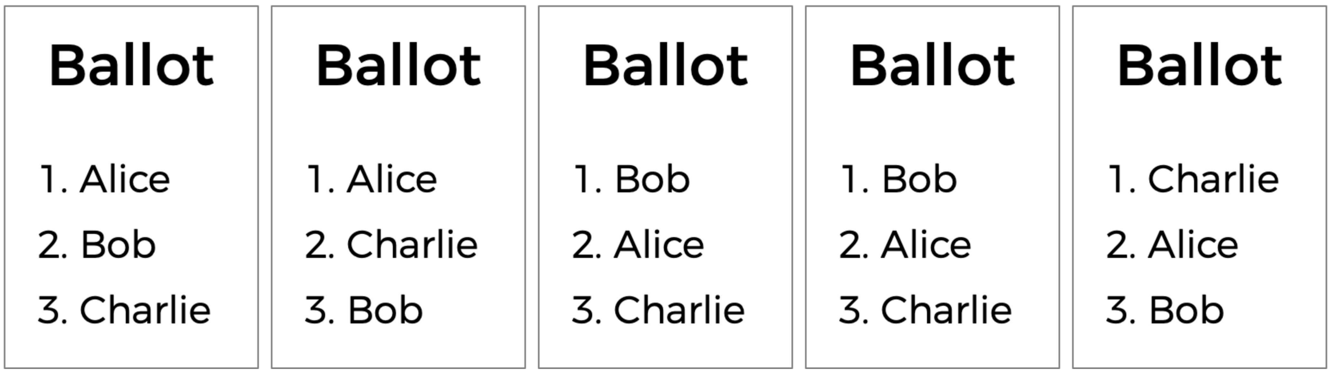
Mas o voto de pluralidade tem algumas desvantagens. O que acontece, por exemplo, em uma eleição com três candidatos, e as cédulas abaixo são lançadas?



Ballot	Ballot	Ballot	Ballot	Ballot
Alice	Alice	Bob	Bob	Charlie

Uma votação de pluralidade aqui declararia um empate entre Alice e Bob, já que cada um tem dois votos. Mas esse é o resultado certo?

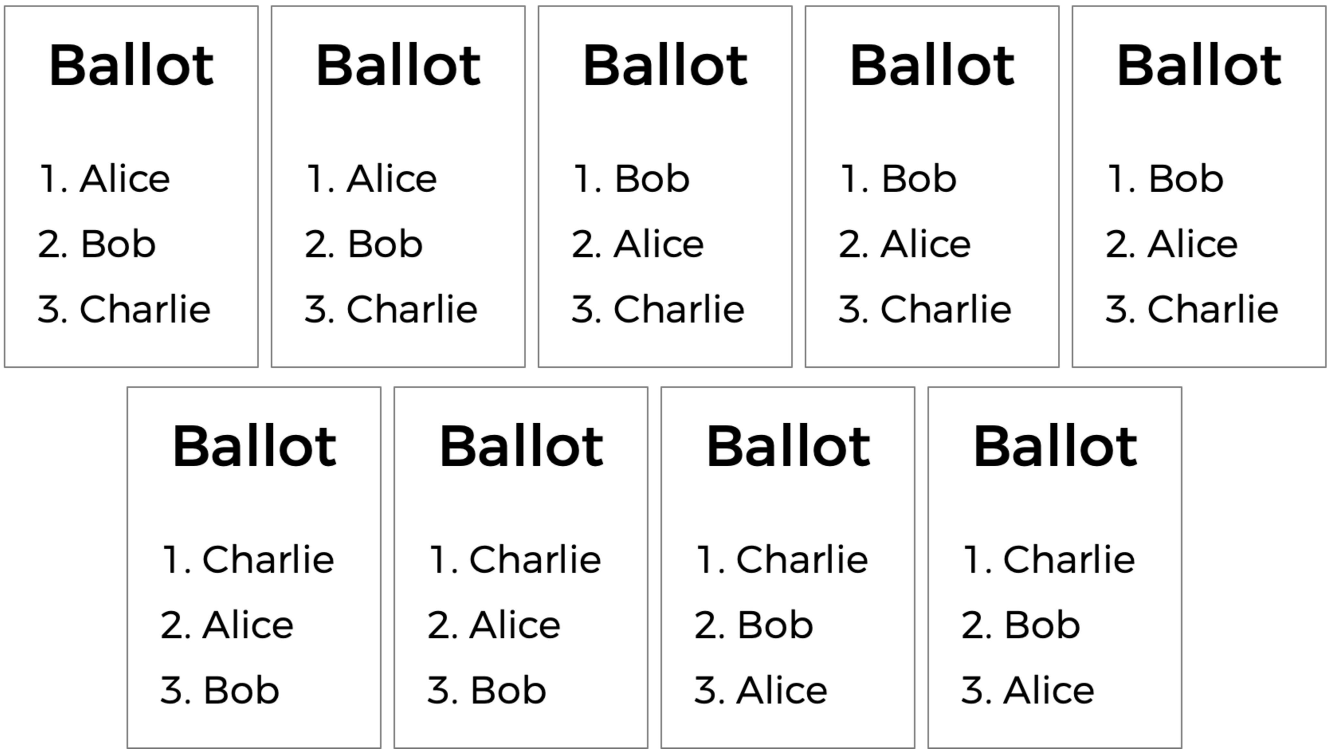
Há outro tipo de sistema de votação, conhecido como sistema de votação por classificação. Em um sistema de escolha ranqueada, os eleitores podem votar em mais de um candidato. Em vez de apenas votar na primeira escolha, eles podem classificar os candidatos em ordem de preferência. As cédulas resultantes podem, portanto, parecer como abaixo.



Ballot	Ballot	Ballot	Ballot	Ballot
1. Alice	1. Alice	1. Bob	1. Bob	1. Charlie
2. Bob	2. Charlie	2. Alice	2. Alice	2. Alice
3. Charlie	3. Bob	3. Charlie	3. Charlie	3. Bob

Here, each voter, in addition to specifying their first preference candidate, has also indicated their second and third choices. And now, what was previously a tied election could now have a winner. The race was originally tied between Alice and Bob, so Charlie was out of the running. But the voter who chose Charlie preferred Alice over Bob, so Alice could here be declared the winner.

Ranked choice voting can also solve yet another potential drawback of plurality voting. Take a look at the following ballots.



Ballot
1. Alice
2. Bob
3. Charlie

Ballot
1. Alice
2. Bob
3. Charlie

Ballot
1. Bob
2. Alice
3. Charlie

Ballot
1. Bob
2. Alice
3. Charlie

Ballot
1. Bob
2. Alice
3. Charlie

Ballot
1. Charlie
2. Alice
3. Bob

Ballot
1. Charlie
2. Alice
3. Bob

Ballot
1. Charlie
2. Bob
3. Alice

Ballot
1. Charlie
2. Bob
3. Alice

Who should win this election? In a plurality vote where each voter chooses their first preference only, Charlie wins this election with four votes compared to only three for Bob and two for Alice. But a majority of the voters (5 out of the 9) would be happier with either Alice or Bob instead of Charlie. By considering ranked preferences, a voting system may be able to choose a winner that better reflects the preferences of the voters.

One such ranked choice voting system is the instant runoff system. In an instant runoff election, voters can rank as many candidates as they wish. If any candidate has a majority (more than 50%) of the first preference votes, that candidate is declared the winner of the election.

If no candidate has more than 50% of the vote, then an "instant runoff" occurrs. The candidate who received the fewest number of votes is eliminated from the election, and anyone who originally chose that candidate as their first preference now has their second preference considered. Why do it this way? Effectively, this simulates what would have happened if the least popular candidate had not been in the election to begin with.

The process repeats: if no candidate has a majority of the votes, the last place candidate is eliminated, and anyone who voted for them will instead vote for their next preference (who hasn't themselves already been eliminated). Once a candidate has a majority, that candidate is declared the winner.

Let's consider the nine ballots above and examine how a runoff election would take place.

Alice has two votes, Bob has three votes, and Charlie has four votes. To win an election with nine people, a majority (five votes) is required. Since nobody has a majority, a runoff needs to be held. Alice has the fewest number of votes (with only two), so Alice is eliminated. The voters who originally voted for Alice listed Bob as second preference, so Bob gets the extra two vote. Bob now has five votes, and Charlie still has four votes. Bob now has a majority, and Bob is declared the winner.

What corner cases do we need to consider here?

One possibility is that there's a tie for who should get eliminated. We can handle that scenario by saying all candidates who are tied for last place will be eliminated. If every remaining candidate has the exact same number of votes, though, eliminating the tied last place candidates means eliminating everyone! So in that case, we'll have to be careful not to eliminate everyone, and just declare the election a tie between all remaining candidates.

Some instant runoff elections don't require voters to rank all of their preferences — so there might be five candidates in an election, but a voter might only choose two. For this problem's purposes, though, we'll ignore that particular corner case, and assume that all voters will rank all of the candidates in their preferred order.

Sounds a bit more complicated than a plurality vote, doesn't it? But it arguably has the benefit of being an election system where the winner of the election more accurately represents the preferences of the voters.

Getting Started

Log into code.cs50.io (https://code.cs50.io/), 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/2022/fall/psets/3/runoff.zip
```

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

Then execute

```
unzip runoff.zip
```

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

```
rm runoff.zip
```

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

Now type

```
cd runoff
```

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

```
runoff/ $
```

If all was successful, you should execute

```
ls
```

and see a file named `runoff.c`. Executing `code runoff.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!

Understanding

Let's take a look at `runoff.c`. We're defining two constants: `MAX_CANDIDATES` for the maximum number of candidates in the election, and `MAX_VOTERS` for the maximum number of voters in the election.

Next up is a two-dimensional array `preferences`. The array `preferences[i]` will represent all of the preferences for voter number `i`, and the integer `preferences[i][j]` here will store the index of the candidate who is the `j`th preference for voter `i`.

Next up is a `struct` called `candidate`. Every `candidate` has a `string` field for their `name`, and `int` representing the number of `votes` they currently have, and a `bool` value called `eliminated` that indicates whether the candidate has been eliminated from the election. The array `candidates` will keep track of all of the candidates in the election.

The program also has two global variables: `voter_count` and `candidate_count`.

Now onto `main`. Notice that after determining the number of candidates and the number of voters, the main voting loop begins, giving every voter a chance to vote. As the voter enters their preferences, the `vote` function is called to keep track of all of the preferences. If at any point, the ballot is deemed to be invalid, the program exits.

Once all of the votes are in, another loop begins: this one's going to keep looping through the runoff process of checking for a winner and eliminating the last place candidate until there is a winner.

The first call here is to a function called `tabulate`, which should look at all of the voters' preferences and compute the current vote totals, by looking at each voter's top choice candidate who hasn't yet been eliminated. Next, the `print_winner` function should print out the winner if there is one; if there is, the program is over. But otherwise, the program needs to determine the fewest number of votes anyone still in the election received (via a call to `find_min`). If it turns out that everyone in the election is tied with the same number of votes (as determined by the `is_tie` function), the election is declared a tie; otherwise, the last-place candidate (or candidates) is eliminated from the election via a call to the `eliminate` function.

If you look a bit further down in the file, you'll see that these functions — `vote`, `tabulate`, `print_winner`, `find_min`, `is_tie`, and `eliminate` — are all left to up to you to complete!

Specification

Complete the implementation of `runoff.c` in such a way that it simulates a runoff election. You should complete the implementations of the `vote`, `tabulate`, `print_winner`, `find_min`, `is_tie`, and `eliminate` functions, and you should not modify anything else in `runoff.c` (and the inclusion of additional header files, if you'd like).

`vote`

Complete the `vote` function.

- The function takes arguments `voter`, `rank`, and `name`. If `name` is a match for the name of a valid candidate, then you should update the global preferences array to indicate that the voter `voter` has

that candidate as their `rank` preference (where `0` is the first preference, `1` is the second preference, etc.).
- If the preference is successfully recorded, the function should return `true`; the function should return `false` otherwise (if, for instance, `name` is not the name of one of the candidates).
- You may assume that no two candidates will have the same name.

▶ **Hints**

`tabulate`

Complete the `tabulate` function.

- The function should update the number of `votes` each candidate has at this stage in the runoff.
- Recall that at each stage in the runoff, every voter effectively votes for their top-preferred candidate who has not already been eliminated.

▶ **Hints**

`print_winner`

Complete the `print_winner` function.

- If any candidate has more than half of the vote, their name should be printed and the function should return `true`.
- If nobody has won the election yet, the function should return `false`.

▶ **Hints**

`find_min`

Complete the `find_min` function.

- The function should return the minimum vote total for any candidate who is still in the election.

▶ **Hints**

`is_tie`

Complete the `is_tie` function.

- The function takes an argument `min`, which will be the minimum number of votes that anyone in the election currently has.
- The function should return `true` if every candidate remaining in the election has the same number of votes, and should return `false` otherwise.

▶ **Hints**

`eliminate`

Complete the `eliminate` function.

- The function takes an argument `min`, which will be the minimum number of votes that anyone in the election currently has.
- The function should eliminate the candidate (or candidates) who have `min` number of votes.

Walkthrough

Usage

Your program should behave per the example below:

```
./runoff Alice Bob Charlie
Number of voters: 5
Rank 1: Alice
Rank 2: Charlie
Rank 3: Bob

Rank 1: Alice
Rank 2: Charlie
Rank 3: Bob

Rank 1: Bob
Rank 2: Charlie
Rank 3: Alice

Rank 1: Bob
Rank 2: Charlie
Rank 3: Alice

Rank 1: Charlie
Rank 2: Alice
Rank 3: Bob

Alice
```

Testing

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

- An election with any number of candidate (up to the `MAX` of `9`)
- Votar em um candidato pelo nome
- Votos inválidos para candidatos que não constam na cédula
- Imprimindo o vencedor da eleição se houver apenas um
- Não eliminar ninguém em caso de empate entre todos os restantes candidatos

Execute o abaixo para avaliar a exatidão do seu código usando `check50`. Mas certifique-se de compilar e testar você mesmo também!

```
check50 cs50/problems/2023/x/runoff
```

Execute o abaixo para avaliar o estilo do seu código usando `style50`.

```
style50 runoff.c
```

Como enviar

Em seu terminal, execute o abaixo para enviar seu trabalho.

```
submit50 cs50/problems/2023/x/runoff
```