Este é o CS50x

OpenCourseWare

Doar (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://orcid.org/0000-0001-5338-2522)
(https://www.quora.com/profile/David-J-Malan)
(https://www.reddit.com/user/davidjmalan) (https://twitter.com/davidjmalan)

Escoamento

Implemente um programa que execute uma eleição de segundo turno, conforme a seguir.

```
./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 pluralistas, que seguem um algoritmo muito simples para determinar o vencedor de uma eleição: cada eleitor ganha um voto e o candidato com mais votos vence.

Mas o voto plural 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 declararia aqui um empate entre Alice e Bob, uma vez que cada um tem dois votos. Mas é esse o resultado certo?

Existe outro tipo de sistema de votação conhecido como sistema de votação por escolha por classificação. Em um sistema de escolha por classificação, 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, ser semelhantes às apresentadas a seguir.

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

Aqui, cada eleitor, além de especificar seu primeiro candidato preferencial, também indicou sua segunda e terceira opções. E agora, o que antes era uma eleição empatada, agora pode ter um

vencedor. A corrida foi originalmente empatada entre Alice e Bob, então Charlie estava fora da corrida. Mas o eleitor que escolheu Charlie preferiu Alice a Bob, então Alice poderia ser declarada vencedora.

A votação por escolha classificada também pode resolver outra desvantagem potencial da votação por pluralidade. Dê uma olhada nas seguintes cédulas.

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

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

Quem deve ganhar esta eleição? Em uma votação de pluralidade em que cada eleitor escolhe apenas sua primeira preferência, Charlie vence esta eleição com quatro votos, em comparação com apenas três para Bob e dois para Alice. Mas a maioria dos eleitores (5 de 9) ficaria mais feliz com Alice ou Bob em vez de Charlie. Ao considerar as preferências classificadas, um sistema de votação pode ser capaz de escolher um vencedor que reflita melhor as preferências dos eleitores.

Uma dessas opções de sistema de votação por classificação é o sistema de runoff instantâneo. Em um segundo turno, os eleitores podem classificar quantos candidatos quiserem. Se algum candidato tiver a maioria (mais de 50%) dos votos da primeira preferência, esse candidato é declarado vencedor da eleição.

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

Here's how to download this problem's "distribution code" (i.e., starter code) into your own CS50 IDE. Log into CS50 IDE (https://ide.cs50.io/) and then, in a terminal window, execute each of the below.

- Navigate to your `pset3` directory that should already exist.
- [illegible] to make (i.e., create) a directory called [illegible] in [illegible]

- Execute `mkdir runoff` to make (i.e., create) a directory called `runoff` in your `pset3` directory.
- Execute `cd runoff` to change into (i.e., open) that directory.
- Execute `wget https://cdn.cs50.net/2020/fall/psets/3/runoff/runoff.c` to download this problem's distribution code.
- Execute `ls`. You should see this problem's distribution code, in a file called `runoff.c`.

Understanding

Let's open up `runoff.c` to take a look at what's already there. 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`.

O próximo é um `struct` chamado `candidate`. Cada `candidate` um tem um `string` campo para seu `name`, e que `int` representa o número de `votes` que possuem atualmente, e um `bool` valor chamado `eliminated` que indica se o candidato foi eliminado da eleição. A matriz `candidates` acompanhará todos os candidatos na eleição.

O programa também possui duas variáveis globais: `voter_count` e `candidate_count`.

Agora em `main`. Observe que, após determinar o número de candidatos e eleitores, o ciclo de votação principal começa, dando a cada eleitor a chance de votar. Conforme o eleitor insere suas preferências, a `vote` função é chamada para controlar todas as preferências. Se a qualquer momento a cédula for considerada inválida, o programa é encerrado.

Uma vez que todos os votos foram alcançados, outro ciclo começa: este vai continuar repetindo o processo de segundo turno para verificar se há um vencedor e eliminar o último candidato a lugar até que haja um vencedor.

A primeira chamada aqui é para uma função chamada `tabulate`, que deve examinar todas as preferências dos eleitores e computar os totais de votos atuais, observando cada candidato eleito pela primeira vez que ainda não foi eliminado. Em seguida, a `print_winner` função deve imprimir o vencedor, se houver; se houver, o programa acabou. Mas, caso contrário, o programa precisa determinar o menor número de votos que alguém ainda na eleição recebeu (por meio de uma ligação para `find_min`). Se for constatado que todos na eleição estão empatados com o mesmo número de votos (conforme determinado pela `is_tie` função), a eleição é declarada empatada; caso contrário, o candidato em último lugar (ou candidatos) é eliminado da eleição por meio de uma chamada para a `eliminate` função.

Se você olhar um pouco mais abaixo no arquivo, você verá que essas funções - `vote`, `tabulate`, `print_winner`, `find_min`, `is_tie`, e `eliminate` - são todos esquerda para até

`tabulate`, `print_winner`, `find_min`, `is_tie`, e `eliminate` são todos esquerda para ate você para completar!

Especificação

Conclua a implementação de de `runoff.c` forma que simule uma eleição de segundo turno. Você deve concluir as implementações dos `vote`, `tabulate`, `print_winner`, `find_min`, `is_tie`, e `eliminate` funções, e você não deve modificar qualquer outra coisa em `runoff.c` (e a inclusão de arquivos de cabeçalho adicionais, se quiser).

`vote`

Conclua a `vote` função.

- A função tem argumentos `voter`, `rank` e `name`. Se `name` for uma correspondência para o nome de um candidato válido, você deve atualizar a matriz de preferências globais para

  indicar que o eleitor `voter` tem aquele candidato como sua `rank` preferência (onde `0` é a primeira preferência, `1` é a segunda preferência, etc.).
- Se a preferência for registrada com sucesso, a função deve retornar `true`; a função deve retornar de `false` outra forma (se, por exemplo, `name` não for o nome de um dos candidatos).
- Você pode presumir que dois candidatos não terão o mesmo nome.

▼ **Dicas**

- Lembre-se de que `candidate_count` armazena o número de candidatos na eleição.
- Lembre-se de que você pode usar `strcmp` (https://man.cs50.io/3/strcmp)para comparar duas strings.
- Recall that `preferences[i][j]` stores the index of the candidate who is the `j` th ranked preference for the `i` th voter.

`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**

- Recall that `voter_count` stores the number of voters in the election.

- Recall that for a voter `i`, their top choice candidate is represented by `preferences[i][0]`, their second choice candidate by `preferences[i][1]`, etc.
- Recall that the `candidate` `struct` has a field called `eliminated`, which will be `true` if the candidate has been eliminated from the election.
- Lembre-se de que o `candidate` `struct` tem um campo chamado `votes`, que provavelmente você desejará atualizar para o candidato preferido de cada eleitor.

`print_winner`

Conclua a `print_winner` função.

- Se algum candidato tiver mais da metade dos votos, seu nome deverá ser impresso em `stdout` e a função deverá retornar `true`.
- Se ninguém ganhou a eleição ainda, a função deve retornar `false`.

▼ **Dicas**

- Lembre-se de que `voter_count` armazena o número de eleitores na eleição. Diante disso, como você expressaria o número de votos necessários para vencer a eleição?

`find_min`

Conclua a `find_min` função.

- A função deve retornar o total mínimo de votos para qualquer candidato que ainda esteja na eleição.

▼ **Dicas**

- Você provavelmente vai querer percorrer os candidatos para encontrar aquele que ainda está na eleição e tem o menor número de votos. Que informações você deve acompanhar enquanto analisa os candidatos?

`is_tie`

Conclua a `is_tie` função.

- A função leva um argumento `min`, que será o número mínimo de votos que alguém na eleição tem atualmente.
- A função deve retornar `true` se todos os candidatos restantes na eleição tiverem o mesmo número de votos, e deve retornar `false` caso contrário.

▼ **Dicas**

- Lembre-se de que o empate ocorre se todos os candidatos ainda na eleição tiverem o

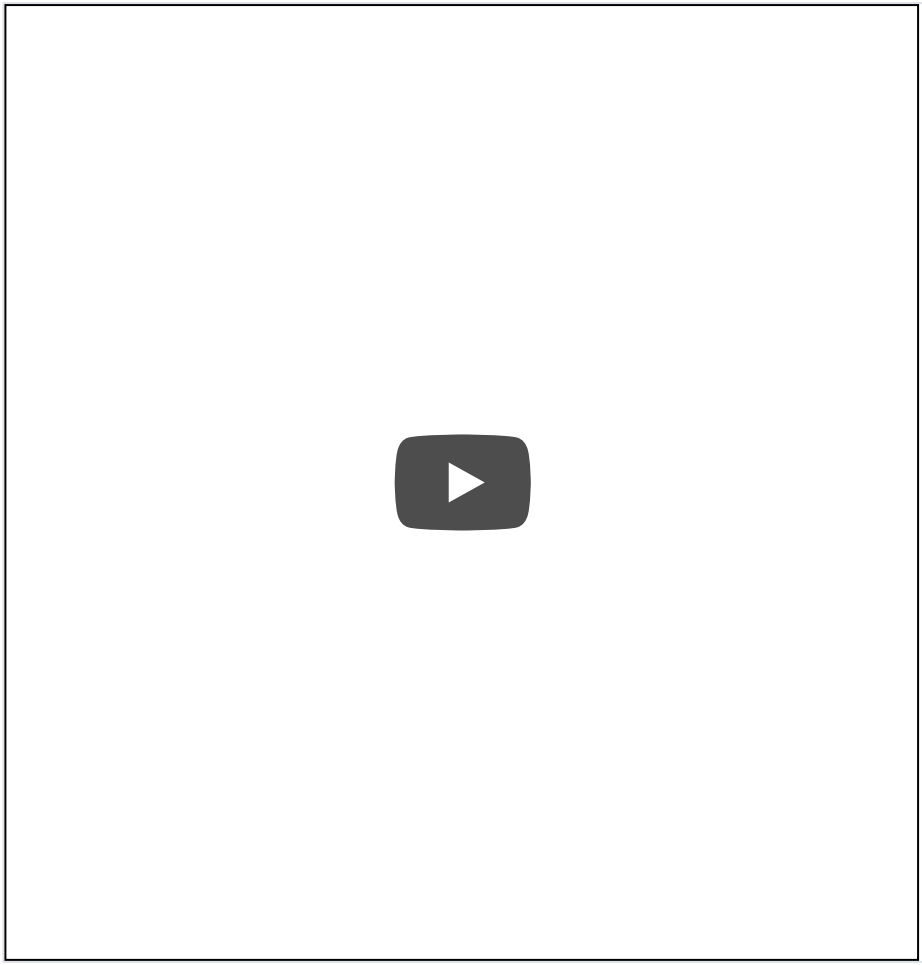
mesmo número de votos. Observe também que a `is_tie` função leva um argumento `min`, que é o menor número de votos que qualquer candidato possui atualmente. Como você pode usar essas informações para determinar se a eleição é um empate (ou, inversamente, não um empate)?

`eliminate`

Conclua a `eliminate` função.

- A função leva um argumento `min`, que será o número mínimo de votos que alguém na eleição tem atualmente.
- A função deve eliminar o candidato (ou candidatos) com `min` número de votos.

Passo a passo



Uso

Seu programa deve se comportar conforme o exemplo abaixo:

```
./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
```

Testando

Certifique-se de testar seu código para certificar-se de que ele lida com ...

- Uma eleição com qualquer número de candidatos (até o `MAX` de `9`)
- Votar em um candidato pelo nome
- Votos inválidos para candidatos que não estão na cédula
- Imprimir o vencedor da eleição se houver apenas um
- Não eliminando ninguém em caso de empate entre todos os candidatos restantes

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

```
check50 cs50/problems/2021/x/runoff
```

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

```
style50 runoff.c
```

Como enviar

Execute o procedimento a seguir, fazendo login com seu nome de usuário e senha do GitHub quando solicitado. Por segurança, você verá asteriscos (`*`) em vez dos caracteres reais em sua senha.

```
submit50 cs50/problems/2021/x/runoff
```