Eleições Majoritárias (Plurality)

Implemente um programa que simula uma eleição do tipo *plurality*

```
$ ./plurality Alice Bob Charlie
Number of voters: 4
Vote: Alice
Vote: Bob
Vote: Charlie
Vote: Alice
Alice
```

Introdução

Eleições podem ter vários tamanhos e formatos. No Reino Unido, UK, o Primeiro Ministro é, oficialmente, considerado como monarca, geralmente escolhido do partido político que teve o maior número de assentos no **House Commons**. Os Estados Unidos usam um Colégio Eleitoral em um processo multi-passos onde o cidadão vota em como cada estado deveria agrupar os eleitores que por sua vez elegem o presidente.

Talvez, o modo mais simples de executar uma eleição seja o método conhecido como "*plurality vote*". No modo *plurality*, cada eleitor vota em um candidato. No final, cada candidato que tem o maior número de votos é declarado vencedor da eleição.

Começando

Faça o *download* do código-base (*distribution code*) logando diretamente no seu CS50 IDE e, em um terminal, executando os comandos abaixo:

- Execute `cd` para garantir que você está na pasta `~/`.
- Execute `mkdir pset3` para criar um diretório chamado `pset3` dentro do seu diretório `home`.
- Execute `cd pset3` para entrar na pasta `pset3`.
- Execute `make plurality` para criar um diretório chamado `plurality` dentro do `pset3`.
- Execute `cd plurality` para entrar no diretório recém criado.
- Execute (`wget https://cdn.cs50.net/2019/fall/psets/3/plurality/plurality.c`) para fazer o *download* do código-base.
- Execute `ls`. Você deverá ver um arquivo chamado `plurality.c`.

Entendendo

Vamos dar uma olhada no arquivo `plurality.c` e detalhar algumas de suas linhas.

A linha `#define MAX 9` é uma sintaxe usada em C para definir constantes. No caso, `MAX` é o nome da constante e seu valor é definido como `9`. Esta constante será usada em seu programa. Ela define o número

máximo de candidatos na eleição.

O arquivo define uma `struct` chamada `candidato`. Cada `candidato` tem dois campos: uma `string` chamada `name` que representa o nome do canditado e um `int` chamado `votes` que representa o número de votos do candidato. A seguir, encontramos uma **variável global** chamada `candidates` que é um *array* de `candidate`.

Agora, vamos à a função `main`. Primeiro, a variável `cadidate_count` é definida de acordo com o número de parâmetros de linha de comando. Em seguida, o programa copia os nomes dos candidatos, fornecidos via linha de comando, para o vetor de candidatos e inicializa o número de votos para zero. Então o programa solicita o número de eleitores. A seguir, o programa solicita que cada eleitor informe seu voto digitando o nome do candidato. Finalmente, a função `main` chama a função `print_winner` que deverá exibir o vencedor (ou vencedores) da eleição.

Se você olhar um pouco mais abaixo no arquivo, irá notar que as funções `vote` e `print_winner` estão em branco.

Especificação

Complete a implementação do `plurality.c` de modo que o programa simule uma eleição majoritária.

- Complete a função `vote`
  - `vote` recebe um único argumento, uma `string` chamada `name`, representando o nome do candidato que recebeu o voto.
  - Se `name` coincide com um dos nomes dos candidatos, o número de votos do candidato é atualizado. A função `vote` deveria retornar `true` indicando que a "cédula de voto" é válida.
  - Se `name` não coincide com nenhum dos candidatos, o total de votos não muda e a função `vote` deveria retornar `false` indicando que a "cédula de voto" é inválida.
  - Você pode assumir que não existem dois candidatos com o mesmo nome.
- Complete a função `print_winner`
  - Esta função deverá exibir o nome do candidato que venceu a eleição e então exibir uma nova linha.
  - É possível que no fim da eleição, existam dois ou mais vencedores. Neste caso, você deveria exibir os nomes de todos os vencedores, cada um em uma linha separada.

Usando o programa

Seu programa deveria se comportar como os exemplos abaixo:

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

$ ./plurality Alice Bob
```

```
Number of voters: 3
Vote: Alice
Vote: Charlie
Invalid vote.
Vote: Alice
Alice

$ ./plurality Alice Bob Charlie
Number of voters: 5
Vote: Alice
Vote: Charlie
Vote: Bob
Vote: Bob
Vote: Alice
Alice
Bob
```

Testando

Realize alguns testes para garantir que seu código funciona

- Teste eleições com um até 9 candidatos.
- Votar em um candidato usando o nome dele.
- Votar para um candidato que não existe.
- Exibir o nome do vencedor quando há apenas um vencedor na eleição.
- Exibir os nomes dos vencedores quando houver múltiplos vencedores.

Execute o comando abaixo para verificar a **corretude** do seu código. Tenha certeza de que o código compila antes de fazer o teste

```
check50 cs50/problems/2020/x/plurality
```

Execute o comando abaixo para verificar a estilização do seu código

```
style50 plurality.c
```

Enviando seu código

Execute o comando abaixo para enviar seu código. Você deverá efetuar *login* com sua credenciais do GitHub. Por questões de segurança, asteriscos (*) aparecesão no lugar dos caracteres da sua senha

```
submit50 cs50/problems/2020/x/plurality
```