Pluralidade é um exercício de introdução bastante simples para o próximo no conjunto de problemas, ou você escolhe a rota menos confortável com Runoff ou a mais confortável com Tideman.

Na Pluralidade, precisamos organizar uma eleição simples. A função principal e os cabeçalhos são fornecidos pelo curso como um código de distribuição. Não podemos alterar o código de distribuição ou falhamos nos testes de equipe. Nossa tarefa é implementar a função que atualiza a contagem de votos para cada candidato e a função que imprime o vencedor, ou vencedores, pois esse modelo simples de eleição permite empates frequentes. Aqui está um exemplo de como o programa deve funcionar.

```
$ ./plurality Alice Bob Charlie
Número de eleitores: 4
Voto: Alice
Voto: John
Voto inválido.
Voto: Charlie
Voto: Alice
Alice
```

Nossos candidatos são solicitados como argumentos de linha de comando quando iniciamos o programa. O código de distribuição se encarrega de perguntar o número de eleitores e depois cada um de seus votos. Vamos dar uma olhada em como ele se parece.

```
#include <cs50.h|
#include <stdio.h|
#include <string.h|// Número máximo de candidatos
#define MAX 9// Candidatos têm nome e contagem de votos
typedef struct
{
    string name;
    votos int;
}
candidato;// Matriz de candidatos
candidatos candidatos[MAX];// Número de candidatos
int candidate_count;// Protótipos de funções
bool vote(string name);
void print_winner(void);int main(int argc, string argv[])
{
    // Verifica se há uso inválido
```

```
    if (argc < 2)
    {
        printf("Uso: plurality [candidate ...]\n");
        retorno 1;
    }// Preencher array de candidatos
    candidato_conta = argc - 1;
    if (candidate_count | MAX)
    {
        printf("O número máximo de candidatos é %i\n", MAX);
        retorno 2;
    }
    for (int i = 0; i < candidato_count; i++)
    {
        candidatos[i].name = argv[i + 1];
        candidatos[i].votos = 0;
    }int voter_count = get_int("Número de eleitores: ");// Faz um loop sobre
todos os votantes
    for (int i = 0; i < voter_count; i++)
    {
        string name = get_string("Vote: ");// Verifica se há voto inválido
        if (!vote(name))
        {
            printf("Voto inválido.\n");
        }
    }// Mostra o vencedor da eleição
    print_winner();
}
```

Bastante auto-explicativo com seus comentários, apenas alguns pontos sobre os quais gostaria de escrever. Temos uma nova ferramenta neste código chamada estrutura. Sua sintaxe é mostrada na linha 9 typedef struct. Estruturas são uma forma de armazenar diversas variáveis que estão relacionadas à mesma coisa em um novo tipo criado por nós mesmos, como int ou char, mas como quisermos chamá-lo. Neste caso, referem-se aos candidatos. Cada candidato será salvo como um tipo "candidato" com nome e contagem de votos dentro de um array inicializado logo abaixo de nossa definição de struct.

O primeiro loop que vemos dentro do main é aquele que preenche esse array. Ele pega nosso argv[1], e assim por diante, e o salva dentro do array. Sendo argv[1] = Alice, teríamos: candidatos[0].nome = Alice, candidatos[0].votes = 0; Tanto Alice quanto 0 fazem parte do mesmo índice dentro do array, e agora podemos acessar uma string e um int usando o mesmo índice! Isso é inestimável para construir nossas funções vote e print_winner .

Para vote, main itera através de cada eleitor pedindo seu voto e então verifica se é um voto válido, ou seja, corresponde a um nome de candidato. Nosso primeiro trabalho é codificar essa verificação. A função recebe um único argumento chamado name, que é o nome do candidato que o eleitor deseja apoiar. Ele deve adicionar um voto a essa contagem de votos do candidato se o voto for válido ou retornar uma declaração de erro se o código for inválido. Ambas as opções manterão o programa em execução posteriormente. Podemos fazer isso com um loop simples.

```
bool vote(nome da string)
{
    for (int n = 0; n < contagem_candidato; n++)
    {
        if (strcmp(nome, candidatos[n].nome) == 0)
        {
            candidatos[n].votos++;
            retorne verdadeiro;
        }
    }
    return false;
}
```

O loop percorre cada candidato verificando se o nome corresponde. Aqui usamos uma função da biblioteca de strings chamada strcmp . É a maneira mais simples de comparar duas strings e retorna 0 se ambas forem iguais. Passada a verificação, o loop adiciona 1(++) à contagem de votos desse candidato, salva em candidatos[índice].votos. Caso contrário, se o loop não encontrar correspondência para o nome, ele retornará false e main se encarregará de imprimir a mensagem de erro.

Podemos fazer alguns testes aqui e ver se a função está retornando a mensagem de erro para votos inválidos.

Em seguida, para a função vencedora de impressão. Precisamos de um contador onde definiremos a maior contagem de votos e precisamos garantir que, se houver empate, ambos ou mais candidatos sejam impressos.

```
void print_winner(void)
{
    // TODO
    int vencedor = 0;
    for (int v = 0; v < contagem_candidato; v++)
```

```
    {
        if (candidatos[v].votos > vencedor)
        {
            vencedor = candidatos[v].votos;
        }
    }
```

Cuidamos do primeiro problema com um loop que verifica os votos de cada candidato e atualiza um inteiro chamado vencedor toda vez que encontra uma contagem maior. Agora sabemos quantos votos o vencedor tem, mas não sabemos quais candidatos têm essa contagem de votos.

```
for (int w = 0; w < candidato_count; w++)
    {
        if (candidates[w].votes == vencedor)
        {
            printf("%s\n", candidatos[w].name);
        }
    }
}
```

Um segundo loop (um loop irmão, não aninhado) resolve isso verificando cada candidato para a mesma contagem de votos e imprimindo-a toda vez que as contagens coincidem. Isso encerra nossa função print_winner . É também o fim do nosso código.

Podemos ter certeza de que está funcionando executando pequenos testes com dois a quatro candidatos.
Abaixo você encontra os testes da equipe para este problema e o código completo com comentários.

```c
// Everything else the functions is a distribution code provided by CS50x.

#include <cs50.h>
#include <stdio.h>
#include <string.h>

// Max number of candidates
#define MAX 9

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

// Array of candidates
candidate candidates[MAX];

// Number of candidates
int candidate_count;

// Function prototypes
bool vote(string name);
void print_winner(void);
```

```c

int main(int argc, string argv[])
{
    // Check for invalid usage
    if (argc < 2)
    {
        printf("Usage: plurality [candidate ...]\n");
        return 1;
    }

    // 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;
    }

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

    // Loop over all voters
    for (int i = 0; i < voter_count; i++)
    {
        string name = get_string("Vote: ");

        // Check for invalid vote
        if (!vote(name))
        {
            printf("Invalid vote.\n");
        }
    }

    // Display winner of election
    print_winner();
}

```

```c
//Functions I coded:

// Update vote totals given a new vote
bool vote(string name)
{
    // Iterates through each candidate to see if name matches, then add a vote
    for (int n = 0; n < candidate_count; n++)
    {
        if (strcmp(name, candidates[n].name) == 0)
        {
            candidates[n].votes++;
            return true;
        }
    }
    return false;
}

// Print the winner (or winners) of the election
void print_winner(void)
{
    // TODO
    int winner = 0;
    // Updates the highest vote count
    for (int v = 0; v < candidate_count; v++)
    {
        if (candidates[v].votes > winner)
        {
            winner = candidates[v].votes;
        }
    }
    // Print all candidates that match the highest vote count
    for (int w = 0; w < candidate_count; w++)
    {
        if (candidates[w].votes == winner)
        {
            printf("%s\n", candidates[w].name);
        }
    }
}
```