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

// Máximo de Candidatos e Eleitores.
#define MAX_VOTERS 100
#define MAX_CANDIDATES 9

// preferences[i][j] serve para tabular a preferência de cada eleitor.
int preferences[MAX_VOTERS][MAX_CANDIDATES];

// Candidate é uma estrutura que contem um nome, quantidade de votos e o
status do candidato (eliminado ou não).
typedef struct
{
    string name;
    int votes;
    bool eliminated;
}
candidate;

// Array de Candidatos, não podendo ser maior que o máximo de candidatos.
candidate candidates[MAX_CANDIDATES];

// Variáveis para controle do número de candidatos e eleitores(votos).
int voter_count;
int candidate_count;

// Protótipos de funções.
bool vote(int voter, int rank, string name);
void tabulate(void);
bool print_winner(void);
int find_min(void);
bool is_tie(int min);
void eliminate(int min);

int main(int argc, string argv[])
{
    // Verifica se o usuário digitou ao menos um candidato.
    if (argc < 2)
    {
        printf("Usage: runoff [candidate ...]\n");
        return 1;
    }

    // Verifica se o número de candidatos digitados cabe no array.
    candidate_count = argc - 1;
    if (candidate_count > MAX_CANDIDATES)
    {
```

```
        printf("Maximum number of candidates is %i\n", MAX_CANDIDATES);
        return 2;
    }
    // Preenche o nome de cada candidato em um local do array, deixando
os votos deles zerados e todos na disputa.
    for (int i = 0; i < candidate_count; i++)
    {
        candidates[i].name = argv[i + 1];
        candidates[i].votes = 0;
        candidates[i].eliminated = false;
    }

    // Solicita o nº de eleitores(votos) e verifica se está dentro do
limite.
    voter_count = get_int("Number of voters: ");
    if (voter_count > MAX_VOTERS)
    {
        printf("Maximum number of voters is %i\n", MAX_VOTERS);
        return 3;
    }

    // Continua solicitando votos, até atingir o nº de eleitores
digitados.
    for (int i = 0; i < voter_count; i++)
    {

        // Pergunta o rank de cada candidato, repete até o número de
candidatos.
        for (int j = 0; j < candidate_count; j++)
        {
            string name = get_string("Rank %i: ", j + 1);

            // Retorna que o voto é inválido caso o usuário digite um
nome fora da disputa.
            if (!vote(i, j, name))
            {
                printf("Invalid vote.\n");
                return 4;
            }
        }

        printf("\n");
    }

    // Continua rodando "runoff" até que exista um vencedor
    while (true)
    {
        // Calcula a quantidade de votos dos candidatos na disputa
        tabulate();
```

```
        // Verifica se teve algum vencedor
        bool won = print_winner();
        if (won)
        {
            break;
        }

        // Verifica qual é o menor número de votos.
        int min = find_min();

        // Verifica se todos os candidatos ainda na disputa estão
empatados.
        bool tie = is_tie(min);

        // Se houver empate, imprime o nome de todos os candidatos ainda
na disputa.
        if (tie)
        {
            for (int i = 0; i < candidate_count; i++)
            {
                if (!candidates[i].eliminated)
                {
                    printf("%s\n", candidates[i].name);
                }
            }
            break;
        }

        // Elimina os candidatos que tem o menor número de votos.
        eliminate(min);

        // Reseta os votos para recomeçar a verificação.
        for (int i = 0; i < candidate_count; i++)
        {
            candidates[i].votes = 0;
        }
    }
    return 0;
}

// Salva as preferências de cada eleitor
bool vote(int voter, int rank, string name)
{

    // Verifica cada candidato.
    for(int i = 0; i < candidate_count; i++)
    {
```

```
        // Se o nome do candidato for compatível com um existente na
competição, salva o mesmo na posição escolhida.
        // voter = Eleitor votante; Rank = Preferência do eleitor.
        if(strcmp(candidates[i].name, name) == 0)
        {
            preferences[voter][rank] = i;
            return true;
        }
    }
    return false;
}

// Conta os votos de acordo com os candidatos ainda na disputa.
void tabulate(void)
{
    int j = 0;

    // Verifica voto a voto.
    for(int i = 0; i < voter_count;  i++)
    {
        // Se o candidato já não estiver na competição, passa para a
próxima opção deste eleitor.
        while(candidates[preferences[i][j]].eliminated)
        {
            j++;
        }
        // Assim que achar um candidato ainda na disputa, soma 1 voto ao
mesmo.
        candidates[preferences[i][j]].votes++;
        j = 0;
    }
    return;
}

// Imprime o vencedor da eleceição.
bool print_winner(void)
{
    // variável para 50% dos votos.
    float half = voter_count/2;

    // Verifica se cada candidato tem mais que 50% dos votos.
    for(int i = 0; i < candidate_count; i++)
    {
        // Se o candidato tiver > 50% imprime ele como vencedor.
        if(candidates[i].votes > half)
        {
            printf("%s\n", candidates[i].name);
            return true;
        }
```

```
    }
    return false;
}

// Procura e retorna o menor número de votos que um candidato tem.
int find_min(void)
{
    // Variável com valor de 50% dos votos para referência.
    float min_vote = voter_count/2;

    // Verifica candidato a candidato.
    for(int i = 0; i < candidate_count; i++)
    {
        if(!candidates[i].eliminated)
        {
            // Se o nº de votos é menor que o valor que corresponde 50%,
salva o mesmo na variável.
            if(candidates[i].votes < min_vote)
            {
                min_vote = candidates[i].votes;
            }
        }
    }

    //Retorna os votos do candidato com o menor número.
    return min_vote;
}

// Verifica se houve empate, retornando verdadeiro, caso contrario
retorna falso.
bool is_tie(int min)
{
    //  Váriaveis de controle.
    int control_cand = candidate_count;// Controla o nº de candidatos na
disputa.
    int control_emp =  0; // Controla o nº de candidatos empatados.

    // Verifica candidato a candidato.
    for(int i = 0; i < candidate_count; i++)
    {
        if(!candidates[i].eliminated) // Verifica se o candidato ainda
esta na disputa
        {
            if(candidates[i].votes == min) // Verifica se o candidato tem
o mesmo valor de votos do candidato com menos votos.
            {
                control_emp++; // Soma 1 a variável de candidatos
empatados.
            }
```

```
        }
        else // Se não estiver na disputa, retira-se 1 da variavel de
candidatos na disputa
        {
            control_cand--;
        }

        // Verifica se o número de candidatos na disputa é igual ao
número de candidatos empatados.
        if(control_cand == control_emp)
        {
            return true;
        }
    }
    return false;
}

// Elimina os candidatos com o menor número de votos.
void eliminate(int min)
{

    // Verifica candidato a candidato.
    for(int i = 0; i < candidate_count; i++)
    {
        // Se ele ainda não foi eliminado da disputa.
        if(!candidates[i].eliminated)
        {
            // Se o número de votos dele é igual ao menor número de
votos.
            if(candidates[i].votes == min)
            {
                // Elimina o mesmo.
                candidates[i].eliminated = true;
            }
        }
    }

    return;
}
```