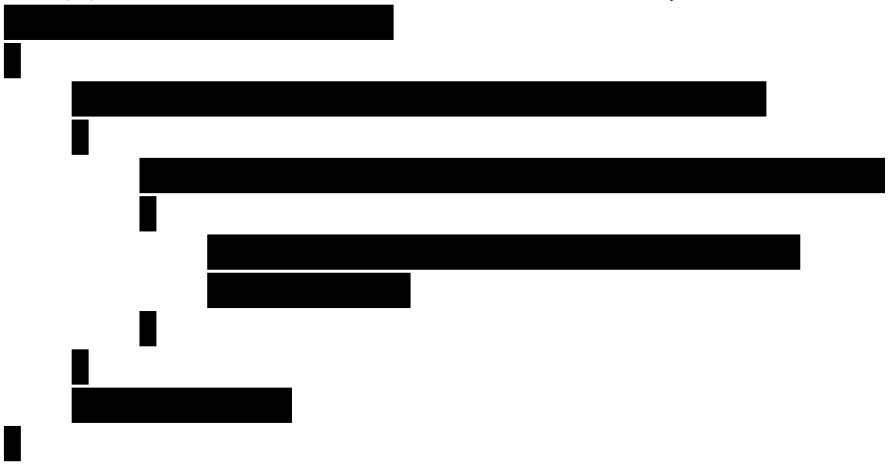
```
// Yannique Hecht
// HARVARD CS50 Week 3 - Runoff - Implement a program that runs a
 runoff election (ranked-choice voting system)

#include <cs50.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
// Define constants (Max voters and candidates)
[redacted]

// Define two-dimensional array (preferences[i][j] is jth preference
 for voter i)
[redacted]

// Define data structure candidates (withe name, vote count,
 eliminated status)
[redacted]
[redacted]
    [redacted]
    [redacted]
    [redacted]
[redacted]
[redacted]

// Create array of candidates
[redacted]

// Define 2 global variables (number of voters and candidates)
[redacted]
[redacted]

// Function prototypes
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]

int main(int argc, string argv[])
[redacted]
    // Check for invalid usage
    [redacted]
    [redacted]
        [redacted]
```

```
    [redacted]
    [redacted]

    // Populate array of candidates. Check for invalid command line
usage and return instructions
    [redacted]
    [redacted]
    [redacted]
    [redacted]
[redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]

    [redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]
    [redacted]

    // Keep querying for votes
    [redacted]
    [redacted]

        // Query for each rank
        [redacted]
        [redacted]
            [redacted]

            // Record vote, unless it's invalid
            [redacted]
            [redacted]
                [redacted]
                [redacted]
            [redacted]
        [redacted]

        [redacted]
    [redacted]
```

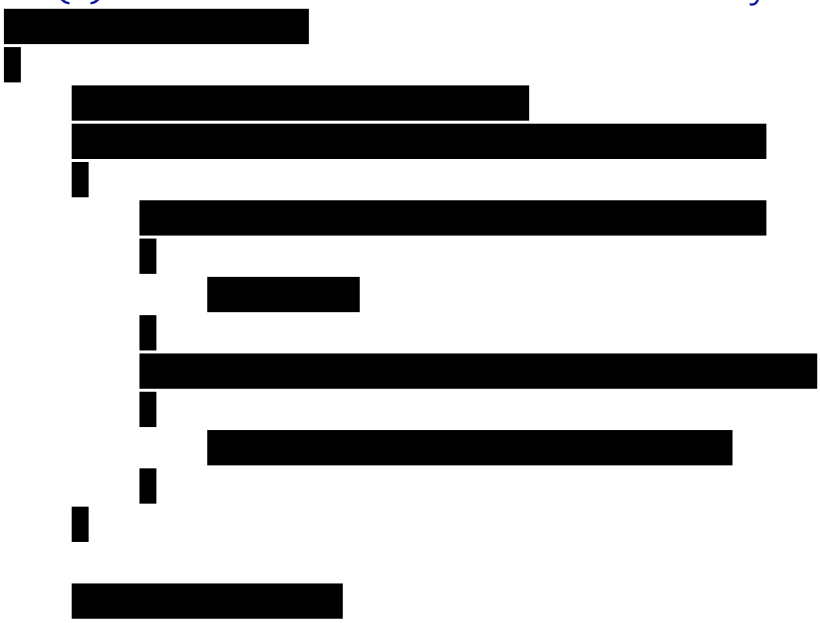
```
    // Keep holding runoffs until winner exists
    [redacted]
    [redacted]
        // Calculate votes given remaining candidates
        [redacted]

        // Check if election has been won
        [redacted]
        [redacted]
        [redacted]
            [redacted]
        [redacted]

        // Eliminate last-place candidates
        [redacted]
        [redacted]

        // If tie, everyone wins
        [redacted]
        [redacted]
            [redacted]
            [redacted]
                [redacted]
                [redacted]
                    [redacted]
                [redacted]
            [redacted]
            [redacted]
        [redacted]

        // Eliminate anyone with minimum number of votes
        [redacted]

        // Reset vote counts back to zero
        [redacted]
        [redacted]
            [redacted]
        [redacted]
    [redacted]
    [redacted]
[redacted]


//PSEUDOCODE: (1) VOTE FUNCTION
// 1. Iterate through candidates
// 2. If name match with candidate, then add vote decision to
```

```
 respective position in array
// 3. Return true, else return false & instructions for use

// (1) Record preference if vote is valid
[redacted]

//PSEUDOCODE: (2) TABULATE FUNCTION
// 1. Iterate through voter count
// 2. Check in which rank
// 3. Check each vote in rank 1 whether eliminated
// 4. If eliminated, go to rank 2
// 5. Check candidate match
// 6. Add 1 vote
// 7. If not eliminated, add 1 vote for candidate from rank 1

// (2) Tabulate votes for non-eliminated candidates
[redacted]

//PSEUDOCODE: (3) PRINT_WINNER FUNCTION
```

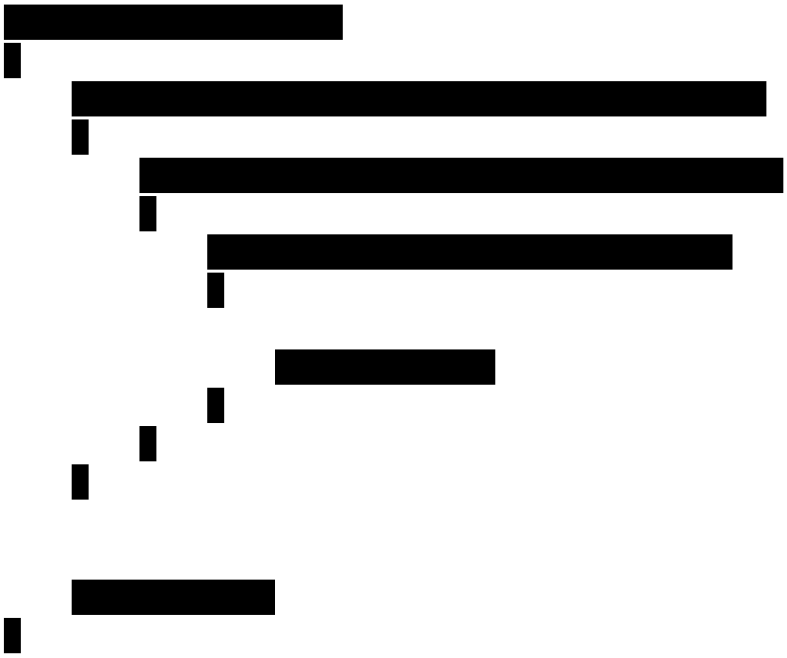
```
// 1.  Divide voter_count by 2
// 2.  Iterate through candidates
// 3.  If x candidate's total votes > (voter_count/2), print
 candidate name
// 4.  Return true
// 5.  Else, return false

// (3) Print the winner of the election, if there is one
```



```
//PSEUDOCODE: (4) FIND_MIN FUNCTION
// 1. Define min vote variable == 1st candidate
// 2. Iterate through candidates
// 3. If y candidate's votes < x candidate's votes, update min vote
// 4. Else, continue
// 5. Return min vote

// (4) Return the minimum number of votes any remaining candidate has
```



```
}


//PSEUDOCODE: (5) IS_TIE FUNCTION
// 1. Iterate through candidates
// 2. Check for not eliminated candidates
// 3. If number of eliminated candidates -1 equal to number of
 candidates, return
// 4. If false, return false
// 5. Else, return, true

// (5) Return true if the election is tied between all candidates,
 false otherwise
```



```


//PSEUDOCODE: (6) ELIMINATE FUNCTION
// 1. Iterate through candidates
// 2. If candidate vote == min number of votes & if candidate not
 eliminated, then eliminate

// (6) Eliminate the candidate (or candidiates) in last place
```