This is CS50

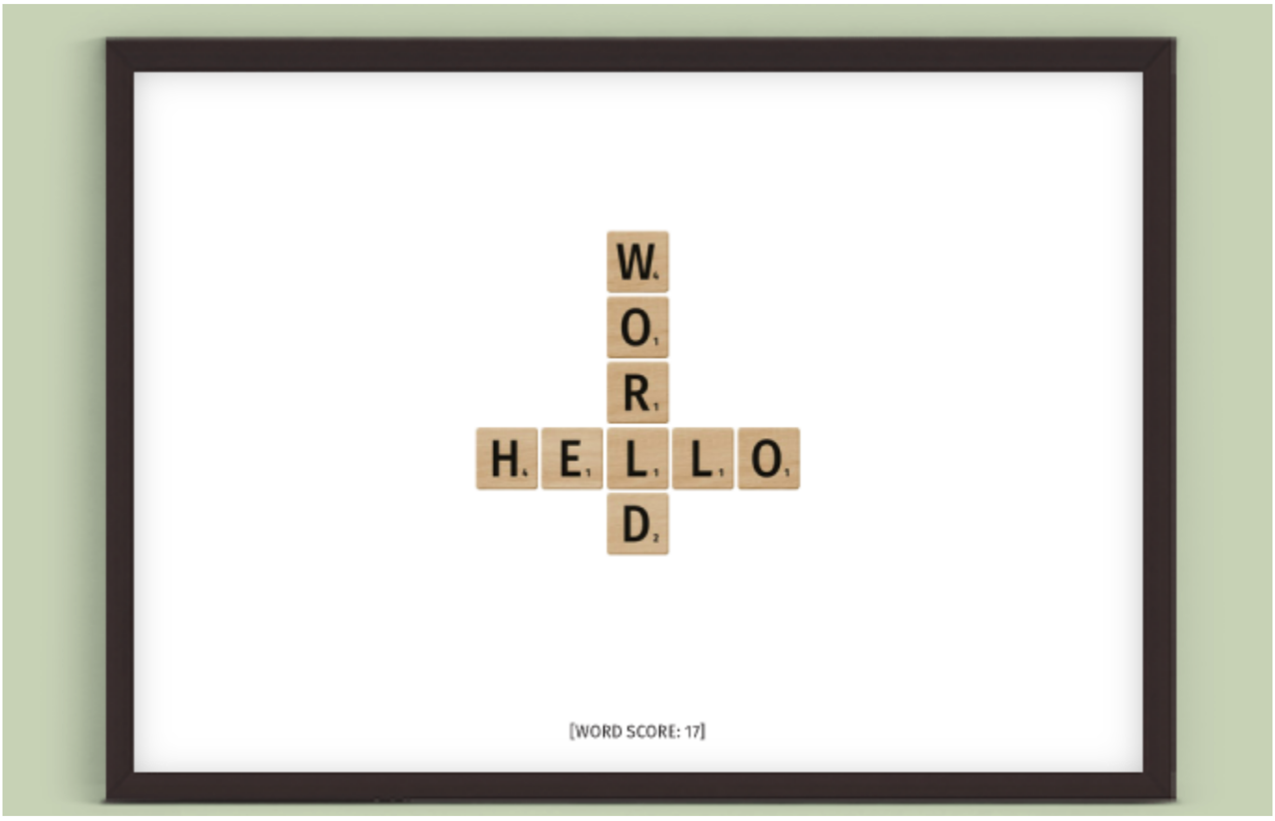
CS50's Introduction to Computer Science

OpenCourseWare

Donate (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://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Scrabble



Problem to Solve

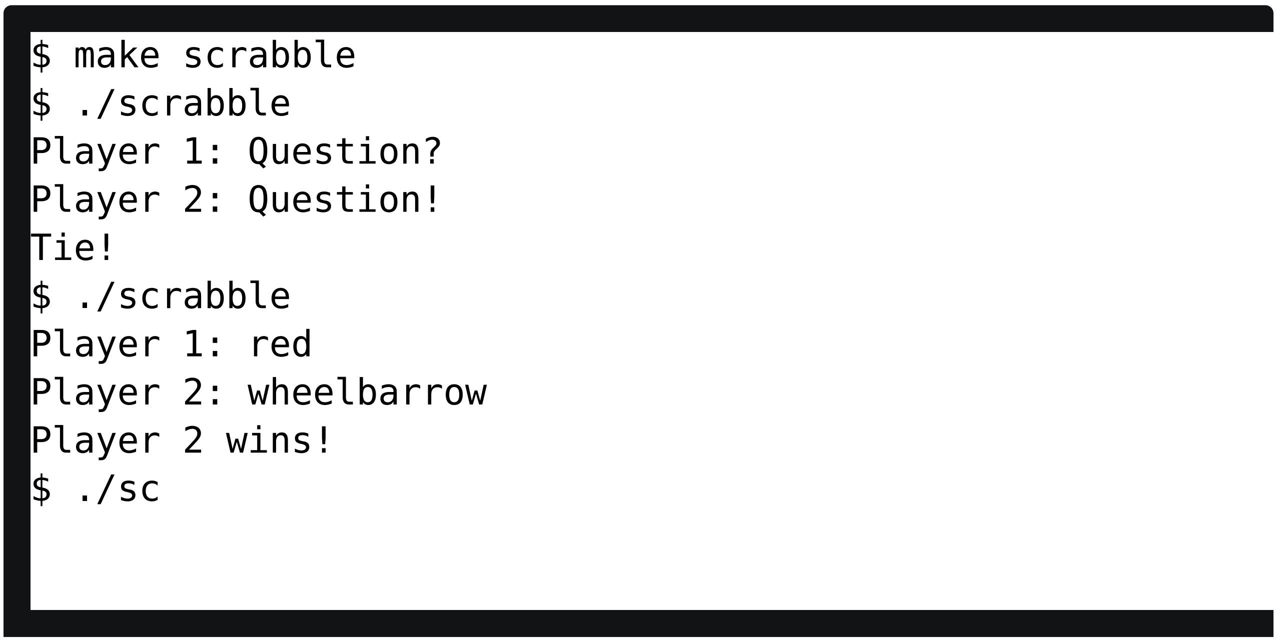
In the game of Scrabble (https://scrabble.hasbro.com/en-us/rules), players create words to score points, and the number of points is the sum of the point values of each letter in the word.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	3	3	2	1	4	2	4	1	8	5	1	3	1	1	3	10	1

For example, if we wanted to score the word "CODE", we would note that the 'C' is worth 3 points, the 'O' is worth 1 point, the 'D' is worth 2 points, and the 'E' is worth 1 point. Summing these, we get that "CODE" is worth 7 points.

In a file called `scrabble.c` in a folder called `scrabble`, implement a program in C that determines the winner of a short Scrabble-like game. Your program should prompt for input twice: once for "Player 1" to input their word and once for "Player 2" to input their word. Then, depending on which player scores the most points, your program should either print "Player 1 wins!", "Player 2 wins!", or "Tie!" (in the event the two players score equal points).

Demo



```
$ make scrabble
$ ./scrabble
Player 1: Question?
Player 2: Question!
Tie!
$ ./scrabble
Player 1: red
Player 2: wheelbarrow
Player 2 wins!
$ ./sc
```

Recorded with asciinema

Advice and Hints

▾ **Write some code that you know will compile**

```
#include <ctype.h>
```

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

int main(void)
{

}
```

Notice that you've now included a few header files that will give you access to functions which might help you solve this problem.

▼ Write some pseudocode before writing more code

If unsure how to solve the problem itself, break it down into smaller problems that you can probably solve first. For instance, this problem is really only a handful of problems:

1. Prompt for the user for two words
2. Compute the score of each word
3. Print the winner

Let's write some pseudcode as comments to remind you to do just that:

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

int main(void)
{
    // Prompt the user for two words

    // Compute the score of each word

    // Print the winner
}
```

Some problems in problem sets, like this one, might contain spoilers (like the next one) that ultimately walk you through the entire solution. While you are permitted to use this code, we really do strongly encourage you to try things out yourself first! The other problems in the problem set won't have this sort of walkthrough, and typically the problem that contains the "full spoiler" is a warm-up version of the bigger problem you'll later need to tackle.

▼ Convert the pseudocode to code

First, consider how you might prompt the user for two words. Recall that `get_string`, a function in the CS50 library, can prompt the user for a string.

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

int main(void)
{
    // Prompt the user for two words
    string word1 = get_string("Player 1: ");
    string word2 = get_string("Player 2: ");

    // Compute the score of each word

    // Print the winner
}
```

Next consider how to compute the score of each word. Since the same scoring algorithm applies to both words, you have a good opportunity for *abstraction*. Here we'll define a function called `compute_score` that takes a string, called `word`, as input, and then returns `word`'s score as an `int`.

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

int compute_score(string word);

int main(void)
{
    // Prompt the user for two words
    string word1 = get_string("Player 1: ");
    string word2 = get_string("Player 2: ");

    // Compute the score of each word
    int score1 = compute_score(word1);
    int score2 = compute_score(word2);

    // Print the winner
}

int compute_score(string word)
{
    // Compute and return score for word
}
```

Now turn to implementing `compute_score`. To compute the score of a word, you need to know the point value of each letter in the word. You can associate letters and their point values with an *array*. Imagine an array of 26 `int`s, called `POINTS`, in which the first number is the point value for 'A', the second number is the point value for 'B', and so on. By declaring and initializing such an array outside of any single function, you can ensure this array is accessible to any function, including `compute_score`.

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

// Points assigned to each letter of the alphabet
int POINTS[] = {1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3, 1, 1, 3, 10, 1, 1,

int compute_score(string word);

int main(void)
{
    // Prompt the user for two words
    string word1 = get_string("Player 1: ");
    string word2 = get_string("Player 2: ");

    // Compute the score of each word
    int score1 = compute_score(word1);
    int score2 = compute_score(word2);

    // Print the winner
}

int compute_score(string word)
{
    // Compute and return score for word
}
```

To implement `compute_score`, first try to find the point value of a single letter in `word`.

- Recall that to find the character at the nth index of a string, `s`, you can write `s[n]`. So `word[0]`, for example, will give you the first character of `word`.
- Now, recall that computers represent characters using ASCII (http://asciitable.com/), a standard that represents each character as a number.
- Recall too that the 0th index of `POINTS`, `POINTS[0]`, gives you the point value of 'A'. Think about how you could transform the numeric representation of 'A' into the index of its point value. Now, what about 'a'? You'll need to apply different transformations to upper- and lower-case letters, so you may find the functions `isupper` (https://manual.cs50.io/3/isupper) and `islower` (https://manual.cs50.io/3/islower) to be helpful to you.
- Keep in mind that characters that are *not* letters should be given zero points For example, `!` is worth 0 points.

If you can properly calculate the value of *one* character in `words`, odds are you can use a loop to sum the points for the rest of the characters. Once you've tried the above on your own, consider this (quite revealing!) hint below.

```
#include <ctype.h>
#include <cs50.h>
#include <stdio.h>
```

```
#include <string.h>

// Points assigned to each letter of the alphabet
int POINTS[] = {1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3, 1, 1, 3, 10, 1, 1,

int compute_score(string word);

int main(void)
{
    // Prompt the user for two words
    string word1 = get_string("Player 1: ");
    string word2 = get_string("Player 2: ");

    // Compute the score of each word
    int score1 = compute_score(word1);
    int score2 = compute_score(word2);

    // Print the winner
}

int compute_score(string word)
{
    // Keep track of score
    int score = 0;

    // Compute score for each character
    for (int i = 0, len = strlen(word); i < len; i++)
    {
        if (isupper(word[i]))
        {
            score += POINTS[word[i] - 'A'];
        }
        else if (islower(word[i]))
        {
            score += POINTS[word[i] - 'a'];
        }
    }

    return score;
}
```

Finally, finish your pseudocode's last step: printing the winner. Recall that an `if` statement can be used to check if a condition is true, and that the additional usage of `else if` or `else` can check for other (exclusive) conditions.

```
if (/* Player 1 wins */)
{
    // ...
}
else if (/* Player 2 wins */)
{
    // ...
}
else
{
    // ...
}
```

And once you've tried the above, feel free to take a peek at the hint (or, rather, complete solution!) below.

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

// Points assigned to each letter of the alphabet
int POINTS[] = {1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3, 1, 1, 3, 10, 1, 1,

int compute_score(string word);

int main(void)
{
    // Prompt the user for two words
    string word1 = get_string("Player 1: ");
    string word2 = get_string("Player 2: ");

    // Compute the score of each word
    int score1 = compute_score(word1);
    int score2 = compute_score(word2);

    // Print the winner
    if (score1 > score2)
    {
        printf("Player 1 wins!\n");
    }
    else if (score1 < score2)
    {
        printf("Player 2 wins!\n");
    }
    else
    {
        printf("Tie!\n");
    }
}

int compute_score(string word)
{
    // Keep track of score
    int score = 0;

    // Compute score for each character
    for (int i = 0, len = strlen(word); i < len; i++)
    {
        if (isupper(word[i]))
        {
            score += POINTS[word[i] - 'A'];
        }
        else if (islower(word[i]))
        {
            score += POINTS[word[i] - 'a'];
        }
    }
    return score;
}
```

How to Test

Your program should behave per the examples below.

```
$ ./scrabble
Player 1: Question?
Player 2: Question!
Tie!
```

```
$ ./scrabble
Player 1: red
Player 2: wheelbarrow
Player 2 wins!
```

```
$ ./scrabble
Player 1: COMPUTER
Player 2: science
Player 1 wins!
```

```
$ ./scrabble
Player 1: Scrabble
Player 2: wiNNeR
Player 1 wins!
```

Correctness

In your terminal, execute the below to check your work's correctness.

```
check50 cs50/problems/2024/x/scrabble
```

Style

Execute the below to evaluate the style of your code using `style50`.

```
style50 scrabble.c
```

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2024/x/scrabble
```