Mario - CS50x 2022

cs50.harvard.edu/x/2022/psets/1/mario/less/

Mario

Getting Started

Open VS Code.

Start by clicking inside your terminal window, then execute `cd` by itself. You should find that its “prompt” resembles the below.

```
$
```

Click inside of that terminal window and then execute

```
wget https://cdn.cs50.net/2021/fall/psets/1/mario-less.zip
```

followed by Enter in order to download a ZIP called `mario-less.zip` in your codespace. Take care not to overlook the space between `wget` and the following URL, or any other character for that matter!

Now execute

```
unzip mario-less.zip
```

to create a folder called `mario-less`. You no longer need the ZIP file, so you can execute

```
rm mario-less.zip
```

and respond with “y” followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd mario-less
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
mario-less/ $
```

If all was successful, you should execute

```
ls
```

and see a file named `mario.c`. Executing `code mario.c` should open the file where you will type your code for this problem set. If not, retrace your steps and see if you can determine where you went wrong!

World 1-1

Toward the end of World 1-1 in Nintendo's Super Mario Brothers, Mario must ascend right-aligned pyramid of blocks, a la the below.

Let's recreate that pyramid in C, albeit in text, using hashes (`#`) for bricks, a la the below. Each hash is a bit taller than it is wide, so the pyramid itself will also be taller than it is wide.

```
       #
      ##
     ###
    ####
   #####
  ######
 #######
########
```

The program we'll write will be called `mario`. And let's allow the user to decide just how tall the pyramid should be by first prompting them for a positive integer between, say, 1 and 8, inclusive.

Here's how the program might work if the user inputs `8` when prompted:

```
$ ./mario
Height: 8
       #
      ##
     ###
    ####
   #####
  ######
 #######
########
```

Here's how the program might work if the user inputs `4` when prompted:

```
$ ./mario
Height: 4
   #
  ##
 ###
####
```

Here's how the program might work if the user inputs `2` when prompted:

```
$ ./mario
Height: 2
 #
##
```

And here's how the program might work if the user inputs `1` when prompted:

```
$ ./mario
Height: 1
#
```

If the user doesn't, in fact, input a positive integer between 1 and 8, inclusive, when prompted, the program should re-prompt the user until they cooperate:

```
$ ./mario
Height: -1
Height: 0
Height: 42
Height: 50
Height: 4
   #
  ##
 ###
####
```

How to begin? Let's approach this problem one step at a time.

Walkthrough

Watch Video At: https://youtu.be/NAs4FIWkJ4s

Pseudocode

First, execute

```
cd
```

to ensure you're in your codespace's default directory.

Then, execute

```
cd mario-less
```

to change to your `mario-less` directory.

Then, execute

```
code pseudocode.txt
```

to open the file called `pseudocode.txt` inside that directory.

Write in `pseudocode.txt` some pseudocode that implements this program, even if not (yet!) sure how to write it in code. There's no one right way to write pseudocode, but short English sentences suffice. Recall how we wrote pseudocode for finding someone in a phone book. Odds are your pseudocode will use (or imply using!) one or more functions, conditionals, Boolean expressions, loops, and/or variables.

▶ Spoiler

Prompting for Input

Whatever your pseudocode, let's first write only the C code that prompts (and re-prompts, as needed) the user for input. Open the file called `mario.c` inside of your `mario` directory. (Remember how?)

Now, modify `mario.c` in such a way that it prompts the user for the pyramid's height, storing their input in a variable, re-prompting the user again and again as needed if their input is not a positive integer between 1 and 8, inclusive. Then, simply print the value of that variable, thereby confirming (for yourself) that you've indeed stored the user's input successfully, a la the below.

```
$ ./mario
Height: -1
Height: 0
Height: 42
Height: 50
Height: 4
Stored: 4
```

▶ Hints

Building the Opposite

Now that your program is (hopefully!) accepting input as prescribed, it's time for another step.

It turns out it's a bit easier to build a left-aligned pyramid than right-, a la the below.

```
#
##
###
####
#####
######
#######
########
```

So let's build a left-aligned pyramid first and then, once that's working, right-align it instead!

Modify `mario.c` at right such that it no longer simply prints the user's input but instead prints a left-aligned pyramid of that height.

▶ Hints

Right-Aligning with Dots

Let's now right-align that pyramid by pushing its hashes to the right by prefixing them with dots (i.e., periods), a la the below.

```
.......#
......##
.....###
....####
...#####
..######
.#######
########
```

Modify `mario.c` in such a way that it does exactly that!

▶ Hint

How to Test Your Code

Does your code work as prescribed when you input

- `-1` (or other negative numbers)?
- `0` ?
- `1` through `8` ?
- `9` or other positive numbers?
- letters or words?
- no input at all, when you only hit Enter?

Removing the Dots

All that remains now is a finishing flourish! Modify `mario.c` in such a way that it prints spaces instead of those dots!

How to Test Your Code

Execute the below to evaluate the correctness of your code using `check50`. But be sure to compile and test it yourself as well!

```
check50 cs50/problems/2022/x/mario/less
```

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

```
style50 mario.c
```

▶ Hint

How to Submit

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

```
submit50 cs50/problems/2022/x/mario/less
```