Problem 1-8: Mario

This is CS50. Harvard University. Fall 2014.

Table of Contents

Questions? Feel free to head to CS50 on Reddit[1], CS50 on StackExchange[2], or the CS50 Facebook group[3].

Objectives

- Get further practice with loops and taking user input.
- Use terminal printing to communicate information to a user graphically.

Recommended Reading

- Pages 1 – 7, 9, and 10 of http://www.howstuffworks.com/c.htm.
- Chapters 1 – 5, 9, and 11 – 17 of *Absolute Beginner's Guide to C*.
- Chapters 1 – 6 of *Programming in C*.

Academic Honesty

This course's philosophy on academic honesty is best stated as "be reasonable." The course recognizes that interactions with classmates and others can facilitate mastery of

[1] https://www.reddit.com/r/cs50
[2] http://cs50.stackexchange.com
[3] https://www.facebook.com/groups/cs50

the course’s material. However, there remains a line between enlisting the help of another and submitting the work of another. This policy characterizes both sides of that line.

The essence of all work that you submit to this course must be your own. Collaboration on problems is not permitted (unless explicitly stated otherwise) except to the extent that you may ask classmates and others for help so long as that help does not reduce to another doing your work for you. Generally speaking, when asking for help, you may show your code or writing to others, but you may not view theirs, so long as you and they respect this policy’s other constraints. Collaboration on quizzes and tests is not permitted at all. Collaboration on the final project is permitted to the extent prescribed by its specification.

Below are rules of thumb that (inexhaustively) characterize acts that the course considers reasonable and not reasonable. If in doubt as to whether some act is reasonable, do not commit it until you solicit and receive approval in writing from your instructor. If a violation of this policy is suspected and confirmed, your instructor reserves the right to impose local sanctions on top of any disciplinary outcome that may include an unsatisfactory or failing grade for work submitted or for the course itself.

Reasonable

- Communicating with classmates about problems in English (or some other spoken language).
- Discussing the course’s material with others in order to understand it better.
- Helping a classmate identify a bug in his or her code, such as by viewing, compiling, or running his or her code, even on your own computer.
- Incorporating snippets of code that you find online or elsewhere into your own code, provided that those snippets are not themselves solutions to assigned problems and that you cite the snippets' origins.
- Reviewing past years' quizzes, tests, and solutions thereto.
- Sending or showing code that you’ve written to someone, possibly a classmate, so that he or she might help you identify and fix a bug.
- Sharing snippets of your own solutions to problems online so that others might help you identify and fix a bug or other issue.
- Turning to the web or elsewhere for instruction beyond the course’s own, for references, and for solutions to technical difficulties, but not for outright solutions to problems or your own final project.

- Whiteboarding solutions to problems with others using diagrams or pseudocode but not actual code.
- Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your work for you.

Not Reasonable

- Accessing a solution to some problem prior to (re-)submitting your own.
- Asking a classmate to see his or her solution to a problem before (re-)submitting your own.
- Decompiling, deobfuscating, or disassembling the staff's solutions to problems.
- Failing to cite (as with comments) the origins of code, writing, or techniques that you discover outside of the course's own lessons and integrate into your own work, even while respecting this policy's other constraints.
- Giving or showing to a classmate a solution to a problem when it is he or she, and not you, who is struggling to solve it.
- Looking at another individual's work during a quiz or test.
- Paying or offering to pay an individual for work that you may submit as (part of) your own.
- Providing or making available solutions to problems to individuals who might take this course in the future.
- Searching for, soliciting, or viewing a quiz's questions or answers prior to taking the quiz.
- Searching for or soliciting outright solutions to problems online or elsewhere.
- Splitting a problem's workload with another individual and combining your work (unless explicitly authorized by the problem itself).
- Submitting (after possibly modifying) the work of another individual beyond allowed snippets.
- Submitting the same or similar work to this course that you have submitted or will submit to another.

- Using resources during a quiz beyond those explicitly allowed in the quiz's instructions.
- Viewing another's solution to a problem and basing your own solution on it.

Assessment

Your work on this problem set will be evaluated along four axes primarily.

Scope
To what extent does your code implement the features required by our specification?

Correctness
To what extent is your code consistent with our specifications and free of bugs?

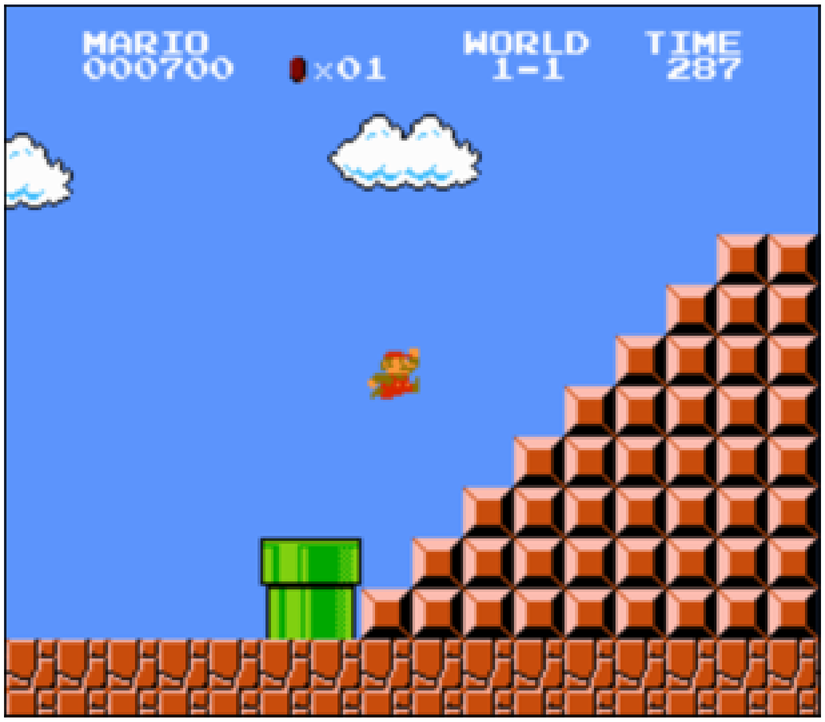
Design
To what extent is your code written well (i.e., clearly, efficiently, elegantly, and/or logically)?

Style
To what extent is your code readable (i.e., commented and indented with variables aptly named)?

To obtain a passing grade in this course, all students must ordinarily submit all assigned problems unless granted an exception in writing by the instructor.

Itsa Mario

Toward the end of World 1-1 in Nintendo's Super Mario Brothers, Mario must ascend a "half-pyramid" of blocks before leaping (if he wants to maximize his score) toward a flag pole. Below is a screenshot.



Write, in a file called `mario.c` in your `~/workspace/unit1` directory, a program that recreates this half-pyramid using hashes (`#`) for blocks. However, to make things more interesting, first prompt the user for the half-pyramid's height, a non-negative integer no greater than `23`. (The height of the half-pyramid pictured above happens to be `8`.) If the user fails to provide a non-negative integer no greater than `23`, you should re-prompt for the same again. Then, generate (with the help of `printf` and one or more loops) the desired half-pyramid. Take care to align the bottom-left corner of your half-pyramid with the left-hand edge of your terminal window, as in the sample output below, wherein underlined text represents some user's input.

```
username@ide50:~/workspace/unit1 $ ./mario
height: 8
       ##
      ###
     ####
    #####
   ######
  #######
 ########
#########
```

Note that the rightmost two columns of blocks must be of the same height. No need to generate the pipe, clouds, numbers, text, or Mario himself.

By contrast, if the user fails to provide a non-negative integer no greater than `23`, your program's output should instead resemble the below, wherein underlined text again represents some user's input. (Recall that `GetInt` will handle some, but not all, re-prompting for you.)

```
username@ide50:~/workspace/unit1 $ ./mario
Height: -2
Height: -1
Height: foo
Retry: bar
Retry: 1
##
```

To compile your program, remember that you can execute

```
make mario
```

or, more manually,

```
clang -o mario mario.c -lcs50
```

after which you can run your program with the below.

```
./mario
```

If you'd like to check the correctness of your program with `check50`, you may execute the below.

```
check50 1516.unit1.mario mario.c
```

And if you'd like to play with the staff's own implementation of mario, you may execute the below.

```
~cs50/unit1/mario
```

Not sure where to begin? Not to worry. A walkthrough awaits!

https://www.youtube.com/watch?v=z32BxNe2Sfc
This was Problem 1-8.