

# Assignment 2 – Hangman Report Template

Your Name

CSE 13S – Winter 24

## Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, “What does this thing do?”. This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

## Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader’s life easier, please do not remove the questions, and simply put your answers below the text of each question. To fill in the answers and edit this file, you can upload the provided zip file to overleaf by pressing [New project] and then [Upload project].

## Guesses

One of the most common questions in the past has been the best way to keep track of which letters have already been guessed. To help you out with this, here are some questions (that you must answer) that may lead you to the best solution.

- How many valid single character guesses are there? What are they?
- Do we need to keep track of how many times something is guessed? Do you need to keep track of the order in which the user makes guesses?
- What data type can we use to keep track of guesses? Keep in mind your answer to the previous questions. Make sure the data type you chose is easily printable in alphabetical order. <sup>1</sup>
- Based on your previous response, how can we check if a letter has already been guessed. <sup>2</sup>

## Strings and characters

- Python has the functions `chr()` and `ord()`. Describe what these functions do. If you are not already familiar with these functions, do some research into them.
- Below is some python code. Finish the C code below so it has the same effect. <sup>3</sup>

```
x = 'q'
print(ord(x))
```

C Code:

---

<sup>1</sup>Your answer should not involve rearranging the old guesses when a new one is made.

<sup>2</sup>The answer to this should be 1-2 lines of code. Keep it simple. If you struggle to do this, investigate different solutions to the previous questions that make this one easier.

<sup>3</sup>Do not hard code any numeric values.

---

```
char x = 'q';
```

- Without using `ctype.h` or **any** numeric values, write C code for `is_uppercase_letter()`. It should return false if the parameter is not uppercase, and true if it is.

```
#include <stdbool.h>
char is_uppercase_letter(char x){

}
```

- What is a string in C? Based on that definition, give an example of something that you would assume is a string that C will not allow.
- What does it mean for a string to be null terminated? Are strings null terminated by default?
- What happens when a program that is looking for a null terminator and does not find it.
- In this assignment, you are given a macro called `CLEAR_SCREEN`. What is its data type? How and when do you use it?

## Testing

List what you will do to test your code. Make sure this is comprehensive.<sup>4</sup> Remember that you will not be getting a reference binary<sup>5</sup>.

## How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, “How do I use this thing?”. Don’t copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

To show “code font” text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

Here is some code in `lstlisting`.

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here’s how.<sup>6</sup>

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this<sup>[1][2][3]</sup>.

## Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, “How is this thing organized so that I can have a chance of fixing it?”. This section will be longer for a more complicated program and shorter for a less complicated program.

---

<sup>4</sup>This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

<sup>5</sup>The output of your binary is not the only thing you should be testing!

<sup>6</sup>This is my footnote.

---

## Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

## Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

## References

- [1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/C\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/C_(programming_language)), 2023. [Online; accessed 20-April-2023].
- [2] Robert Mecklenburg. *Managing Projects with GNU Make*, 3rd ed. O'Reilly, Cambridge, Mass., 2005.
- [3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.