Assignment 2 – Hangman Report Template

Jason Maheru

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.

- This program is a representation of the classic word game, Hangman. The game can be played by many people. There is one person who inputs the secret word or phrase and everyone else guesses letters to find the secret word before they run out of lives.

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. [1]
- Based on your previous response, how can we check if a letter has already been guessed. [2]

1. There are 26 valid single-character guesses. The valid guesses are the 26 letters of the alphabet.
2. Yes we need to keep track of guesses because then that will decide how many lives the person has left depending on how many wrong guesses they made. You have to order the guesses in alphabetical order when they are wrong since the right ones will automatically fill in the blanks in the phrase.
3. I will be using an array to keep track of the guesses. The datatype will be char.
4. I will run a loop that would iterate through the array to check if the letter has already been guessed.

[1] Your answer should not involve rearranging the old guesses when a new one is made.

[2] 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.

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. [3]

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

C Code:

```
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 it's data type? How and when do you use it?

1. Function `chr()` takes an integer representing an ASCII code and returns the corresponding character. The function `ord()` function takes a character, a string of length 1, and returns its ASCII code.
2.
```
x = 'q'
print(ord(x))
```

   C Code:

```
char x = 'q';
print(x);
```

3. c code

```
#include <stdbool.h>
char is_uppercase_letter(char x){
    if (x >= 'A' && x <= 'Z');{
        return True;
    } else{
        return False;
    }
}
```

4. A string is an array of characters followed by a null character, "forwardslash0". From the previous questions char x = 'q', is not a string.
5. It will result in undefined behavior and return an error.
6. Macros usually don't have a data type. Unless assigned. In this case, it is a string data type, that clears the screen when called upon.

[3] Do not hard code any numeric values.

Testing

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

- I will use the text files provided such the the lose.txt and the win.txt to check that my program is operating as it is supposed to.

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.[6]

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[1][2][3].

- The game allows the user to input a secret word or phrase with special characters such as spaces, apostrophes('), and hyphens(-), making the game more dynamic. The user is required to provide the secret as a command line argument before starting the game, with a maximum length of 256 characters(`./hangman "phrase"`). If the secret includes special characters, they are revealed at the beginning and every time the guesses are updated. If the secret comprises only special characters, the player has already won. During the game, the player guesses one letter at a time. If the user fails to enter a secret or provides an incorrect number of arguments, the program will display an error message: `" wrong number of arguments, usage: ./hangman <secret word or phrase>."` It's important to note that when entering a multi-word secret, the user must enclose it in quotes.

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.

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.

[4] 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.

[5] The output of your binary is not the only thing you should be testing!

[6] This is my footnote.

```
remove \. in the functions
#include "hangman\_helpers.h"
bool is\_lowercase\_letter(char c){}

    return only if a letter is lowercase between a thru z.

bool validate\_secret(const char *secret){}

    size_t len_secret = strlen(secret)

    if (strlen(secret) > MAX_LENGTH)
        printf("the secret phrase is over 256 characters\n")
        return false

    for size_t i = 0; run until i < len_secret
        If secret isn't lowercased or has ' ' or '-' or '\''
             the secret is invalid
             return false
    return true

bool string\_contains\_character(const char *s, char c){}

    while (*s != '\0') {
        if (*s == c) {
            return true;
        }
    s++;
    }
    return false

char read\_letter(void){}

    int integer
    char letter

    while true
        print Guess a letter:
        input = getchar

        if letter is lowercase
            letter is char letter
            break;

    return letter;

void display pointer to secret, correct guesses, and incorrect guess and int incorrect
    print clear screen
    print arts[incorrect]
    print new line
    print new line
    print Phrase:

    run for loop for the length of the secret
        if not lowercase or is in correct guesses
            print %c, secret[i]
        else if is a space hyphen or apostrophe
            print those in their places along to phrase
        else
            print underscores for the places of the letters

    print Eliminated and the incorrect guesses
    print a new line
```

```
bool win (secret and guesses)
    bool letters = false

    for length of secret
        if all characters are guessed
        letters is true
        break


    return !letters

function for guesses
    make sure the inputs are letters

    if letter is correct then update the phrase with the correct letter

    if letter is incorrect then up date Eliminated with the incorrect letter

    make a for loop that alphabetizes the incorrect array

int main
    check that the argument is only 2 and not more
        and if it is print why

    secret is argv[1]

    now validate the secret

    initiate char correct and incorrect arrays to MAX_LENGTH setting them to null
    int incorrect to 0

    make while loop that runs while incorrect < LOSING MISTAKES
        display the screen

        run if statement for win fuction
            print you win
            returns 0

        ask guess using read_letter function

        now makeu sure guess is valid by running guess function

    display game screen if u dont win
    print you lost

    return 0
```

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.**

1. bool string contains character(const char* s, char c): Checks if the string s contains s. If it does return true else return false. String as to be less than 256 characters or equal to.
2. char read letter(void): Asks the user for a guess and only takes in one letter from stdin.
3. bool islowercaseletter(char c): Checks if letter is lower case else returns false.
4. bool validatesecret (const char* secret): Takes in the string known as the secret and checks if it is a valid string in order to play the game. If not it will print out invalid character and print another statement stating what should be included.
5. void display(const char *secret, const char *correctguesses, const char *incorrectguesses, int incorrect): This function basically runs the game screen and updates the Phrase and Eliminated and the hangman picture it self after each guess is made.
6. win(const char *secret, const char *guesses): This function checks if you won or not return true or flase.
7. the guess(const char *secret, char guess, char *correctguesses, char *incorrectguesses,int *incorrect): This basically checks if the guess is correct or incorrect and it also makes sure that the same letter is not guessed multiple times.
8. main(int argc, char *argv[]): This runs the whole program where everything is initiated and where the game loops are ran until the user wins or loses. Also checks a valid argc is entered meaning only 2 arugemnts are entered.

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. 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.