Assignment 3 – Hangman Report

Max Ratcliff

CSE 13S – Spring 24

k

1 Purpose

The purpose of this program is to run a game of hangman for the user who will guess one letter at a time until the man is hung or the game is won.

2 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].

2.1 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? **there are 26 valid character guesses representing all the lowercase letters a-z, and 6 guesses until the player loses**
- 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? **we do not need to keep track of how many times something is guessed as long as we keep track of what has been guessed since if its guessed more than one time we promt the user to redo their guess. we also dont need to track the order since the outputed eliminated guesses will be ordered alphabetically**
- 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. **the best data type to keep track of the guesses would be a character array since its easy to itterate through and search and easy to represent as a string for printing. we can initiate a 6 character string and add a compare any new letters to the existing ones in the string to add it in alphabetical order**[1]
- Based on your previous response, how can we check if a letter has already been guessed. **we can run a linear search on the string array which should only take a for loop**[2]

[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.

2.2 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';
print("%d", (int)x)
```

- 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){
    if( (int)x < 65 || (int)x > 90){
        return false
    }
    return true
}
```

- 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. **a string in C is an array of characters ending in a null terminator. "hello\0world" is an example of a not allowed string as there is a null terminator in the middle**
- What does it mean for a string to be null terminated? Are strings null terminated by default? **it means that the last character is '\0'. strings defined with " are automatically null terminated**
- What happens when a program that is looking for a null terminator and does not find it? **it runs off the edge of the char array resulting in an index out of bounds error**

2.3 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, but you do have a file of inputs (`tester.txt`), and two output files (`expected_win.txt` and `expected_lose.txt`). See the Testing section of the assignment PDF for how to use them. **I will make tests to test for the phrases 'zxywvutsrqponmlkjihgfedcba' and 'don't go in empty-handed' comparing to `expected_win.txt` and `expected_lose.txt`**

3 How to Use the Program

to use this program run `./hangman "my secret phrase"` replacing what's in between the double quotes with the phrase you want to play with that is no longer than 256 characters, then the program will prompt you to enter one letter at a time until you either guess the secret phrase or fail to

[3] Do not hard code any numeric values.

[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.

4 Program Design

the main program is run in `hangman.c`, and helper functions relating to character and string manipulation appear in `hangman_helpers.c`

4.1 Overall Pseudocode

`hangman.c`

- **get secret phrase loop until game is won print gallows print special characters check if phrase is only special characters if so win game print eliminated characters get a letter guess from input check if letter appears in secret phrase if so next iteration and reveal the guessed letter if not add it to eliminated characters and increment gallows array index to print next frame check if phrase as been guessed if so win game next loop iteration**

`hangman_helpers.c`

- **check if string contains character loop through index 0 to 256 check if character at index is desired character if so return true check if character at index is null terminator if so return false return false**
- **read letter from input (and handle errors?) get sdtin check is 1 character error check is valid character error return letter**
- **check for lowercase letter make sure letter is lowercase by checking its in the ascii range from [97, 122]**
- **check validity of secret phrase iterate through potential secret from 0 to 256 or untill null-terminator found use helper method to check that the character is a lowercase letter. if character at index is not a lowercase letter check cases of ' ' (ascii: 32), "' (ascii: 44), '-' (ascii: 45) if none of those cases print "invalid character: 'X'" where X is the invalid letter at the index**

4.2 Function Descriptions

bool string_contains_character()

- **Inputs: pointer to a string, character to find**
- **Outputs: true(1)/false(0)**
- **iterate through a string until target character is found, return true if found and false if not found**

char read_letter()

- **Inputs: user input**
- **Outputs: character that user inputed**
- **get a character from the user, check for its validity**

bool is_lowercase_letter()

- **Inputs: character**

- **Outputs: true(1)/false(0)**
- **make sure letter is lowercase by checking its in the ascii range from [97, 122]**

bool is_valid_secret()

- **Inputs: string pointer to secret**
- **Outputs: true(1)/false(0)**
- **iterate through potential secret from 0 to 256 or untill null-terminator found use helper method to check that the character is a lowercase letter. if character at index is not a lowercase letter check cases of ' ' (ascii: 32), "' (ascii: 44), '-' (ascii: 45) if none of those cases print "invalid character: 'X'" where X is the invalid letter at the index**