Assignment 2 – Hangman
Dalal Arafeh
CSE 13S – Winter 24

Purpose

The purpose of this program is to mimic the hangman game. The main functionality revolves around players attempting to guess a hidden phrase by inputting one letter at a time. The program checks each guess against the secret phrase and updates the game state. Correct guesses reveal the letter in the phrase, while incorrect guesses contribute to a cumulative mistake count. If the player successfully uncovers the entire phrase, they win. After six incorrect guesses, the game will terminate.

Questions

Guesses

- **How many valid single character guesses are there? What are they?**

  26 valid single character guesses, one for each letter of the alphabet.

- **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 or the order in which the user makes the guesses. We only need to keep track of the characters that have been guessed like a real-life hangman game.

- **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**

  To keep track of guesses, we could use an array of boolean values that represents each letter of the alphabet. For example, if a player guesses ‘M’, the associated boolean value at the array’s ‘M’ index is true (1). If a player doesn’t guess ‘H’, the associated index in the array will stay false (0).

- **Based on your previous response, how can we check if a letter has already been guessed.**

  To check if a letter has already been guessed, we can check the boolean array to determine if the value of a letter is true or false whenever a player makes a new guess. Like checking the letters guessed on a board in real life. For example, if the boolean value of a letter’s index is false, that indicates the letter has not been guessed yet. This is useful to show the players progress by displaying the alphabet with the guessed letters filled in and the unguessed letters shown as underscores.

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.**
    - chr(): converts an integer to character
    - ord(): converts a single character to an integer

- **Below is some python code. Finish the C code below so it has the same effect.**

```
char x = 'q';
printf("%d", 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>
bool is_uppercase_letter(char x) {
        return x >= 'A' && x <= 'Z';
}
```

- **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 is an array of characters, each represented as a very small integer. It can be visualized as an array, with the memory address of its first character returned when accessed. A "handle" is placed at the outset of the string to distinguish the beginning. A null character (sentinels) is placed at the end of the string to indicate the termination of the string.

- **What does it mean for a string to be null terminated? Are strings null terminated by default?**

  When a string is null terminated that means that we have reached the end of the string. When you start a string with a set of characters in quotes, the compiler terminates by default by putting a special end marker at the end. When reading characters as input without using special string functions, you would need to manually add a null character at the end to make it a proper string.

- **What happens when a program that is looking for a null terminator and does not find it.**

  Without a null terminator, the program could crash. For example, if I am using strlen or printf, the function may keep reading characters beyond the end of the desired string until it stumbles upon a null character. The functions would also not behave as expected. For example, strlen might return an incorrect length.

- **In this assignment, you are given a macro called CLEAR_SCREEN. What is its data type? How and when do you use it?**

```
// Printing this string clears the screen.
#define CLEAR_SCREEN "\033[1;1H\033[2J"
```

This data type is a string. It would be used at the beginning of the game and after each guess to clear the console of the screen.

Testing

- List what you will do to test your code. Make sure this is comprehensive. Remember that you will not be getting a reference binary.
    - Make sure that the secret phrase is displaced correctly
        - Unguessed letters as underscores
        - Guessed letters shown
    - Invalid inputs
        - uppercase letters, numbers, unaccepted symbols like @#
        - char invalid_inputs[]
    - Test if it can handle secret phrases that go over the maximum length limit (257)
    - Check if program is printing out the correct error messages
    - I could also check my code by putting my output and the expected output into separate text files and comparing them using diff

How to Use the Program

- Compile the program using Makefile (make command)
- Execute the program by typing ./hangman
- The game prompts the user to guess letters one at a time.
- The secret can be multiple words (apostrophes, spaces, hyphens revealed at the start of the game)
- ASCII art for hangman is printed with each guess, updating based on the number of incorrect guesses
- The game concludes either when the player successfully guesses all letters of the word or when the maximum number of incorrect guesses is reached (6)

Program Design

- Main: hangman.c, game flow and interactions, game state
- Helper functions:
    - Hangman_helpers.c
        - string_contains_character
        - read_letter
        - validate_secret
        - is_lowercase_letter
- Secret phrase string: storing the phrase the player needs to guess
- Guessed letters array; which letters have been guessed, correct or incorrect
- Hangman ASCII art array

Pseudocode

---

hangman.c:

```
#include “hangman_helpers.h”

void win(char *secret) {
        printf(“You win! The secret phrase was: %s\n”, secret);
}

void lose(char *secret) {
        printf(“You lose! The secret phrase was: %s\n”, secret);
}
void print_eliminate(char *tracking_letters) {
        printf(“\nEliminated: ”);
        for each character from a to z {
                //display eliminated letter
                printf(“%c”, i);
        }
        printf space between eliminated and win/lose statement
}

int main () {
        //check number of arguments is 2
                //variables
                char secret
                secret length = strlen()
                int incorrect attempts
                int guessed letters
                int letters have been guessed correctly in the secret string
                int mistakes = 0
                char alphabet_letters [26]
                //check secret is valid to initialize the game (A secret must be provided, and at
                  most 256)
                        validate_secret == true

                        //while loop of game
                        while (1) {
                                printf(“%s”, CLEAR_SCREEN);
                                //array of ascii art for hangman (mistakes as an index)
                                printf(“%s\n\n”, arts[mistakes]);
                                //make sure aligned with Eliminate
                                printf(“    Phrase: ”);
                                //loop to print right answers and put them in their designated places
```

```
        //how many characters guessed or printed
        guessed_letters = 0;
        //access characters one by one
        for each character at index in secret string{
                //conditions
                If character is special character {
                //can do special character function to call
                        guessed_letters ++;
                        //print special character
                        printf("%c", secret[i]);
                }else if character is a letter {
                        guessed_letters ++;
                        //print character
                        printf("%c", secret[i]);
                }else {
                        //unknown or unguessed character
                        printf("_");
                }
        }
        //elimination in alphabetical order
        eliminate(tracking_letters);
}

//exit game if player made six wrong attempts
if incorrect attempts == LOSING_MISTAKE {
        lose(secret);
        break;
}
//check if player guessed the secret phrase, won
if number of guessed letters is equal to length of secret string (int){
        //all letters in secret string have been guessed correctly
        win(secret);
        break;
}
//condition1: check if guess is lower case
while (1) {
        Call read_letter()
        is_lowercase_letter (read_letter())
        if valid {

        }
}
//check if guess is right to input
        string_contains_character
```

```
                                        if guess is right {
                                                put in phrase
                                        } else {
                                                put it in eliminated
                                        }
                                //secret is invalid
                                else {
                                        return 1;
                                }

}
```

hangman_helpers.c:

```
#include "hangman_helpers.h"

bool is_lowercase_letter(char c) {
        return c >= 'a' && c <= 'z';
}

bool validate_secret(const char *secret) {
        //until null terminator
        for (int i = 0; s[i] != '\0'; i++) {
                //string s contains the character c
                if (s[i] == c) {
                        return True;
                }
        }
        return False;

}

const char* punctuation = space, apostrophe, hyphen.;
bool string_contains_character(const char *s, char c) {
        //each character in string phrase
        for (int i = 0; secret[i] != '\0'; i++) {
                //two conditions: not lowercase, not a space, apostrophe, hyphen.
                if (!is_lowercase_letter(secret[i]) || not allowed punctuation) {
                        printf("Invalid character: '%c'\n", secret[i]);
                        return false;
                }
        }
        return true;
```

```
}

char read_letter(void) {
        char input;
        printf("guess a letter: ");
        scanf("%c", &input);
        return input;
}
```

Function Description

For each function in your program, you will need to explain your thought process.

- void win(char *secret)
    - Inputs: secret
    - Print winner message when player wins the game and will also show the secret phase
- void lose(char *secret)
    - Input: secret
    - Print loser message when player loses the game and will also show the secret phase
- main ()
    - Check if the game has a valid secret phase
    - Game loop
        - Show secret phrase
            - Shows correctly guessed letters
            - Underscore for unguessed letters
        - Check if player guessed all letters (won)
        - Check if player got too many wrong guesses (lost
    - Prompt letter guess
        - Validate if lower case
        - Update whether correct or incorrect
    - If secret phase is invalid, end game with error message