AE1PGA Lab 6

1. Searching an array

Write a function to search an array of integers for a target integer. The function could be implemented iteratively or recursively. For the latter, you'd better pass a start and end parameter to the search function to keep track of the start and end section of the array you are searching in, rather than try to actually remove numbers from the array.

2. Word-guessing game

You are to write a program to implement a simple word guessing game. Given a list of words, 1 word is randomly selected and then the each letter of the word is displayed as underscore characters '_'. The user can guess a letter and if that letter is in the word then the appropriate underscores are replaced with that letter. If the letter is not in the word then the player loses 1 life. The player has 5 lives in total and the game is over when lives reach zero.

The aim of the game is to guess the word. If the user thinks they know the word then, instead of entering an individual character, they can enter an entire word. If the word is correct they win the game. If the word is wrong then they lose a life.

When the program starts, it will prompt to read in an integer which is the number of words in the word list. The program then reads in that number of words. You can assume that all the words in the game have a maximum of 31 characters. After they have all been entered, the program randomly selects a word to use for the next game. (In a future exercise, we will read this list of words from a file; for now we just have the user type them in.)

The game then starts. The player is shown the word with the appropriate number of underscores and is prompted to enter either a word or character guess. The game the proceeds as described above until the player wins or dies. After displaying an appropriate message, the program asks the user if they want to play again. If yes, then a new word is randomly selected from the word list and the game starts again. If no, the program exits.

The words typed in the word list should be single words, only containing lowercase letters 'a' - 'z' and should not contain numbers, punctuation, spaces, etc.

For generating random numbers, you can use the "rand" and "srand" functions to generate random integers. See the book section 5.10, particularly pages 206 and 210, and example program Fig 5.13.

Example I/O:

```
Number of words: 4
Enter word: football
Enter word: player
Enter word: apples
Enter word: incredible

Word: _ _ _ _ _ _

Guess (5 lives): e

Word: _ _ _ _ e _

Guess (5 lives): t

Word: _ _ _ _ e _

Guess (4 lives): heater

Word: _ _ _ _ e _

Guess (3 lives): a

Word: a _ _ _ _ e _

Guess (3 lives): p

Word: a p p _ e _

Guess (3 lives): apples

Correct!
Do you want to player again (y/n)?  n
```

3. Hangman

The word guessing game above is based on the British game Hangman. In this game, the process of playing the game is exactly the same but as well as writing out the current state of the word, the computer should draw a man hanging from a gallows. As the player gets guesses wrong, more and more of the body of the man is drawn. When the entire body is draw, the player has lost all their lives and the game is over. An example can be seen on the Wikipedia page.

Extend your solution to the above task to also draw a man, using text characters, in the same way that you drew squares and triangles using text characters in earlier exercises in the textbook (eg, 2.21, 2.25, 2.27, 3.32, 3.33, 3.39, 4.16, 4.31). You can decide how to split the man into 5 pieces and how to draw him.

End