KOÇ UNIVERSITY

COMP 201 - Spring 2023
Assignment 2 - Heap Management

Assigned: 30 March 2023 23:59, Due: 13 April 2023 23:59

Nafiseh Tofighi (`ntofighi21@ku.edu.tr`) is the lead person for this assignment.

1. Word Guessing Game

Challenge your vocabulary skills with the Word Guessing Game! In this exciting game, you will be shown a series of dashes that represent a secret word. Your task is to guess the letters of the word one by one until you can figure out what the word is. To help you with your guessing, you will have a limited number of attempts which is equal to the length of the secret word. As you progress through the game, the words may become more predictable since some letters are guessed. Think you have what it takes to guess the words? Play the Word Guessing Game now and find out!

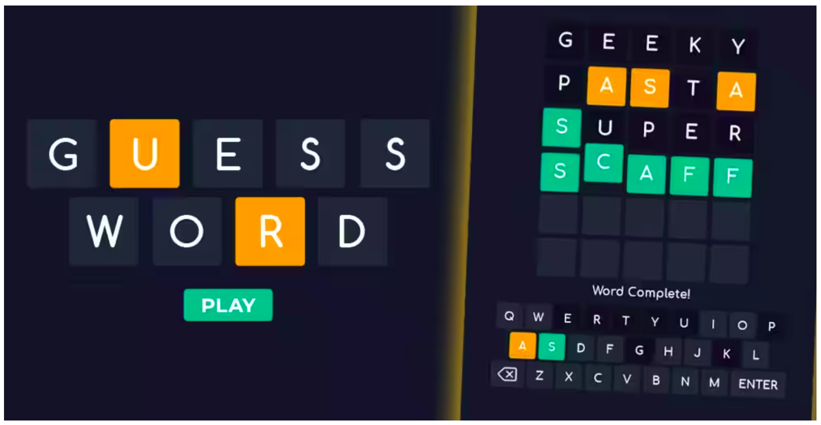


Figure 1

2. Logistics

This is an individual project. All hand-ins are electronic. Clarifications and corrections will be announced on Blackboard.

3. Handout Instructions

3.1 How to start

- Accept the GitHub Classroom assignment using the link: https://classroom.github.com/a/30mw16G5

- Clone the GitHub repository created for you to a Linux machine in which you plan to do your work (We advise you to do your work on our Linux servers [linuxpool.ku.edu.tr]. See Section 8 for details.)

  ```
  $ git clone  https://github.com/COMP-201-Spring-2023/assignment-2-USER.git
  ```

  (Replace USER with your GitHub username that you use to accept the assignment)

- **IMPORTANT** After cloning the repository, you are required to write the following honor code in a new file called `honor.txt` and commit and push it: 'I hereby declare that I have completed this assignment individually, without support from anyone else.'You can use the following command to create `honor.txt` file and write the honor code in it:

  ```
  $ echo "I hereby declare that I have completed this assignment
  individually, without support from anyone else." > honor.txt
  ```

3.2 Main task

You will write your code in the `main.c` file. A template file is provided so that you only have to write the asked functionality in their respective positions. The positions are indicated with comments. The main function is already implemented for you, and you implement seven functions explained in the next section.

A random word will be selected from the `words.txt` file that is provided for you. The player should guess a letter, and the program will detect whether the secret word includes guessed letter or not. The user is allowed to guess the wrong letter according to the length of the secret letter. However, if the player guesses a correct letter, his or her attempts will not be decreed. For example, if the has five remaining attempts and guesses a letter, that secret word includes it. After this guess, the user still has five attempts.

Examples are important to how you implement the functions, and you can use them as test cases for your implementation. **Massages printed in each part must be the same as the examples.** You may use sample input in the main function to test each of implemented functions separately. You have to keep the declaration of the functions, inputs of the functions, and the output of the functions unchanged.

4. Tasks

Please read the instruction carefully and implement the following seven functions in determined positions in `main.c` file.

char get_random_word(char file_name)

Input: The name of the text file that contains list words. In this text file, each word is on a separate line.

Task: Reading words from the text file and selecting one of them randomly. Note that the program should print the appropriate massage if the text file is unavailable. The printed messages must be according to the examples.

Output: The randomly selected word that the user should guess it. The output for this function could be different each time because the words are selected randomly.

Example:

Sample input 1:

```
char* file_name= "words.txt";
char* word = get_random_word(file_name);
printf( "%s", word);
```

Sample output 1:

```
Hurricane
```

Sample input 2:

```
char* file_name= "words.txt";
char* word = get_random_word(file_name);
printf( "%s", word);
```

Sample output 2:

```
Carrot
```

Sample input 3:

```
char* file_name= "wwwords.txt";
char* word = get_random_word(file_name);
printf( "%s", word);
```

Sample output 3:

```
Could not open file wwwords.txt
```

char get_guess()

Task: Function to get a guess from the user. If the user enters more than one letter, the first letter must be considered the guessed character.

Output: The guessed character.

Example:

Sample input 4:

```
char guess;
guess = get_guess();
printf( "%c", guess);
```

Sample output 4:

```
Please enter your guess: a
A
```

Sample input 5:

```
char guess;
guess = get_guess();
printf( "%c", guess);
```

Sample output 5:

```
Please enter your guess: T
T
```

Sample input 6:

```
char guess;
guess = get_guess();
printf( "%c", guess);
```

Sample output 6:

```
Please enter your guess: hat
H
```

void display_word_status(char* word, char guessed_letters[])

Input: The secret word that the user should find.
Letters that were correctly guessed.

Task: Print the current status of the secret word. The dash character should be printed for each letter of the secret word if the user has not guessed it yet. And the letter itself should be printed if it was guessed previously. If a word includes repetitive letters, only a single guess is enough for that word. For example, if the secret word is Hello, the user needs only guess the letter L once, and both L will be printed at the output.

Example:

Sample input 7:

```
char guessed_letters[] ={'T', 'R'};
char* word= "Robot";
display_word_status(word, guessed_letters);
```

Sample output 7:

```
Word: R---T
```

Sample input 8:

```
char guessed_letters[] ={'\0'};
char* word= "Robot";
display_word_status(word, guessed_letters);
```

Sample output 8:

```
Word: -----
```

Sample input 9:

```
char guessed_letters[] ={'S', 'U', 'L'};
char* word= "Robot";
display_word_status(word, guessed_letters);
```

Sample output 9:

```
Word: -----
```

Sample input 10:

```
char guessed_letters[] ={'T', 'R', 'O', 'B'};
char* word= "Robot";
display_word_status(word, guessed_letters);
```

Sample output 10:

```
Word: ROBOT
```

Sample input 11:

```
char guessed_letters[] ={'O'};
char* word= "Robot";
display_word_status(word, guessed_letters);
```

Sample output 11:

```
Word: -O-O-
```

int is_word_guessed(char* word, char guessed_letters[], int guessed_count)

Input: The secret word that the user should guess.
Letters that were correctly guessed.

Task: Function to determine if the word has been fully guessed. In other words, this function will check whether the user has won with the current guess.

Output: 1 if the user has guessed all letters of the secret word.
0 if all letters have yet to be guessed.

Example:

Sample input 12:

```
char guessed_letters[] ={'O'};
char* word= "Robot";
int result= is_word_guessed(word, guessed_letters);
printf("%d", result);
```

Sample output 12:

```
0
```

Sample input 13:

```
char guessed_letters[] ={'O', 'T', 'R', 'B'};
char* word= "Robot";
int result= is_word_guessed(word, guessed_letters);
printf("%d", result);
```

Sample output 13:

```
1
```

int is_guess_valid(char* word, char guessed_letters[], char guess)

Input: The secret word that the user should guess.
Letters that were correctly guessed. The current guess

Task: Function to determine if a guess is valid (not already guessed and in the word.) The guess is invalid if either the character is not a letter or the character previously has been guessed correctly. A valid guess could be in the secret word or not. Note that the printed messages in this function must be according to the examples for each situation.

Output: -1 if the guessed character is not in the secret word
0 if the guessed character is invalid
1 if the guessed character is in the secret word

Example:

Sample input 14:

```
char guessed_letters[] ={'O', 'T', 'R', 'B'};
char* word= "Robot";
char guess= 'a';
int result= is_guess_valid(word, guessed_letters, guess);
printf("%d", result);
```

Sample output 14:

```
The word does not contain a
-1
```

Sample input 15:

```
char guessed_letters[] ={'O', 'T'};
char* word= "Robot";
char guess= 'T';
int result= is_guess_valid(word, guessed_letters, guess);
printf("%d", result);
```

Sample output 15:

```
Invalid guess. You already guessed T
0
```

Sample input 16:

```
char guessed_letters[] ={'O', 'T'};
char* word= "Robot";
char guess= '3';
int result= is_guess_valid(word, guessed_letters, guess);
printf("%d", result);
```

Sample output 16:

```
Invalid guess. Please enter a letter.
0
```

Sample input 17:

```
char guessed_letters[] ={'O', 'T'};
char* word= "Robot";
char guess= 'R';
int result= is_guess_valid(word, guessed_letters, guess);
printf("%d", result);
```

Sample output 17:

```
1
```

int guess_word(char* word)

Input: The secret word that the user should guess.

Task: Function to play the word guessing game. In this function, the previously defined functions (display_word_status, get_guess, is_guess_valid and is_word_guessed) should be called. In addition, the 'Game over'and 'Win'situations should be determined in this function. The user can have a wrong guess based on the length of the secret word. For example, if the secret word is 'Robot', the user may have five wrong guesses. As a result, the player has a limited number of attempts. Note that the printed messages in this function must be according to the examples in each state.

Output: 0 if the player loses the game.
1 if the player guesses the word correctly.

Example:

Sample input 18:

```
char* word= "Robot";
int result= guess_word(word);
printf("%d", result);
```

Sample output 18:

```
Welcome to the word guessing game! The word has 5 letters.
You have 5 guesses left.
Word: -----
Please enter your guess: h
The word does not contain H
You have 4 guesses left.
Word: -----
Please enter your guess: e
The word does not contain E
You have 3 guesses left.
Word: -----
Please enter your guess: s
The word does not contain S
You have 2 guesses left.
Word: -----
Please enter your guess: t
You have 2 guesses left.
Word: ----T
Please enter your guess: u
The word does not contain U
You have 1 guesses left.
Word: ----T
Please enter your guess: o
You have 1 guesses left.
Word: -O-OT
Please enter your guess: g
The word does not contain G
You ran out of guesses. Game over.
The word was Robot.
0
```

Sample input 19:

```
char* word= "Robot";
int result= guess_word(word);
printf("%d", result);
```

Sample output 19:

```
Welcome to the word guessing game! The word has 5 letters.
You have 5 guesses left.
Word: -----
Please enter your guess: a
The word does not contain A
You have 4 guesses left.
Word: -----
Please enter your guess: o
You have 4 guesses left.
Word: -O-O-
Please enter your guess: i
The word does not contain I
You have 3 guesses left.
Word: -O-O-
Please enter your guess: b
You have 3 guesses left.
Word: -OBO-
Please enter your guess: r
You have 3 guesses left.
Word: ROBO-
Please enter your guess: b
You already guessed B
You have 3 guesses left.
Word: ROBO-
Please enter your guess: t
Congratulations, you guessed the word!
The word was Robot.
1
```

void play_game(char* file_name)

Input: The name of the text file that contains list words. In this text file, each word is on a separate line.

Task: In this function, a new random word should be selected by calling the get_random_word function at each iteration. Then with the help of the guess_word function, the user can play the game, and with respect to the output of the guess_word function, the player could either win or lose. Then the program should ask the user whether they want to continue or not. If the user enters yes, both get_random_word and guess_word functions must be called again for the new game round. This procedure continues until the user enters that they do not want to continue the game. Then it should be printed that the total number of rounds and the number of rounds in which the user win the game should. Finally, print all played words, and correctly guessed words should print with the * sign.

Example:

Sample input 20:

```
char* file_name = "words.txt";
play_game(file_name);
```

Sample output 20:

```
Welcome to the word guessing game! The word has 5 letters.
You have 5 guesses left.
Word: -----
Please enter your guess: a
You have 5 guesses left.
Word: ---A-
Please enter your guess: i
The word does not contain I
You have 4 guesses left.
Word: ---A-
Please enter your guess: u
The word does not contain U
You have 3 guesses left.
Word: ---A-
Please enter your guess: o
The word does not contain O
You have 2 guesses left.
Word: ---A-
Please enter your guess: e
You have 2 guesses left.
Word: --EA-
Please enter your guess: r
You have 2 guesses left.
Word: -REA-
Please enter your guess: g
The word does not contain G
You have 1 guesses left.
Word: -REA-
Please enter your guess: m
You have 1 guesses left.
Word: -REAM
Please enter your guess: e
You already guessed E
You have 1 guesses left.
Word: -REAM
Please enter your guess: d
Congratulations, you guessed the word!
The word was Dream.
Do you want to play again?(Y/N)
y
Welcome to the word guessing game! The word has 5 letters.
```

```
You have 5 guesses left.
Word: -----
Please enter your guess: a
Invalid guess. Please enter a letter.
You have 5 guesses left.
Word: -----
Please enter your guess: e
The word does not contain E
You have 4 guesses left.
Word: -----
Please enter your guess: i
You have 4 guesses left.
Word: -I---
Please enter your guess: o
The word does not contain O
You have 3 guesses left.
Word: -I---
Please enter your guess: r
The word does not contain R
You have 2 guesses left.
Word: -I---
Please enter your guess: d
The word does not contain D
You have 1 guesses left.
Word: -I---
Please enter your guess: b
The word does not contain B
You ran out of guesses. Game over.
The word was Pizza.
Do you want to play again?(Y/N)
y
Welcome to the word guessing game! The word has 9 letters.
You have 9 guesses left.
Word: ---------
Please enter your guess: a
Invalid guess. Please enter a letter.
You have 9 guesses left.
Word: ---------
Please enter your guess: e
You have 9 guesses left.
Word: ---E----E
Please enter your guess: i
You have 9 guesses left.
Word: -I-E----E
Please enter your guess: u
The word does not contain U
You have 8 guesses left.
```

```
Word: -I-E----E
Please enter your guess: o
The word does not contain O
You have 7 guesses left.
Word: -I-E----E
Please enter your guess: s
The word does not contain S
You have 6 guesses left.
Word: -I-E----E
Please enter your guess: m
The word does not contain M
You have 5 guesses left.
Word: -I-E----E
Please enter your guess: n
You have 5 guesses left.
Word: -INE----E
Please enter your guess: p
You have 5 guesses left.
Word: PINE-PP-E
Please enter your guess: a
You have 5 guesses left.
Word: PINEAPP-E
Please enter your guess: l
Congratulations, you guessed the word!
The word was Pineapple.
Do you want to play again?(Y/N)
y
Welcome to the word guessing game! The word has 9 letters.
You have 9 guesses left.
Word: ---------
Please enter your guess: a
Invalid guess. Please enter a letter.
You have 9 guesses left.
Word: ---------
Please enter your guess: o
You have 9 guesses left.
Word: ---O--O--
Please enter your guess: e
You have 9 guesses left.
Word: ---O--O-E
Please enter your guess: u
The word does not contain U
You have 8 guesses left.
Word: ---O--O-E
Please enter your guess: l
You have 8 guesses left.
Word: --LO--O-E
```

```
Please enter your guess: sm
The word does not contain S
You have 7 guesses left.
Word: --LO--O-E
Please enter your guess: m
The word does not contain M
You have 6 guesses left.
Word: --LO--O-E
Please enter your guess: d
The word does not contain D
You have 5 guesses left.
Word: --LO--O-E
Please enter your guess: s
The word does not contain S
You have 4 guesses left.
Word: --LO--O-E
Please enter your guess: g
The word does not contain G
You have 3 guesses left.
Word: --LO--O-E
Please enter your guess: p
You have 3 guesses left.
Word: --LOP-O-E
Please enter your guess: j
The word does not contain J
You have 2 guesses left.
Word: --LOP-O-E
Please enter your guess: x
You have 2 guesses left.
Word: X-LOP-O-E
Please enter your guess: z
The word does not contain Z
You have 1 guesses left.
Word: X-LOP-O-E
Please enter your guess: b
The word does not contain B
You ran out of guesses. Game over.
The word was Xylophone.
Do you want to play again?(Y/N)
n
You played 4 times and won 2 of them.
Words:
1)*Dream
2) Pizza
3)*Pineapple
4) Xylophone
```

5. Evaluation

Your score will be computed out of a maximum of 100 points based on the following distribution:

- 15 points: get_random_word function
- 10 points: get_guess function
- 10 points: display_word_status function
- 10 points: is_word_guessed function
- 10 points: is_guess_valid function
- 15 points: play_game function
- 20 points: guess_word function
- 5 points: Style points(Meaningful variables and effective comments)
- 5 points: Effective use of version control tool.

Task Points: Your exact outputs will be matched to some selected test cases (all are different from examples). Therefore, to ease auto grading you must match your outputs to the outputs in the examples. You will get points separately from each of the test cases. So, you can get partial points.

Effective Use of Version Control Points: You are required to push your changes to the repository frequently. If you only push the final version, even if it is implemented 100% correctly, you will lose a fraction of the grade because you are expected to learn to use Version Control Systems effectively. You do not have to push every small piece of change to Github but every meaningful change should be pushed. For example, each of the functions coded and tested can be one commit. **For each function, there should be at least one commit (with proper commit message) that includes just modifications on that task.**

Style Points: Finally, we've reserved 5 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.

Important Note: We use automated plagiarism detection to compare your assignment submission with others and also the code repositories on GitHub and similar sites. Moreover, we plan to ask randomly selected 10% of students to explain their code verbally after the assignments are graded. And one may lose full credit if he or she fails this oral part.

6. Handin Instructions

As with previous assignments, we use GitHub for the submissions as follows. Note that we want you to get used to using a version management system (Git) in terms of writing good commit messages and frequently committing your work so that you can get most out of Git.

- Commit all the changes you make:

```
$ git commit -a -m "commit message"
```

Note: Use meaningful commit messages as in huge projects they become really helpful. Try to gain this habit from early on.

- Push your work to GitHub servers:

```
$ git push origin main
```

7. How to use linuxpool.ku.edu.tr linux servers

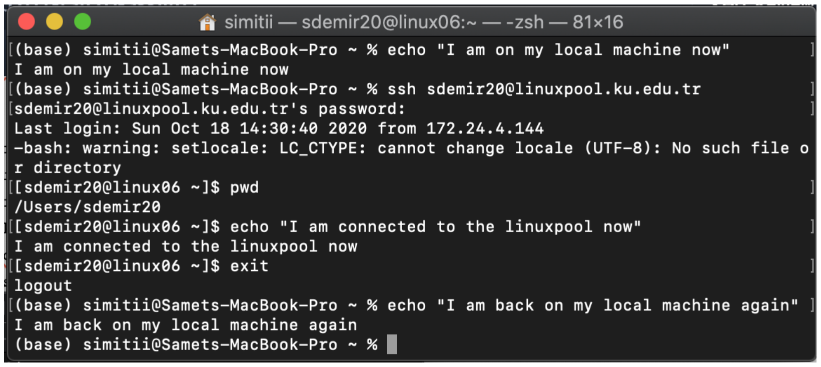
I Connect to KU VPN (If you are connected to the KU network, you can skip this step.)
See for details: https://confluence.ku.edu.tr/kuhelp/ithelp/it-services/network-and-wireless/vpn-access

II Connect to linuxpool.ku.edu.tr server using SSH (Replace USER with your Koç University username):
`$ ssh USER@linuxpool.ku.edu.tr`
(It will ask your password, type your Koç University password.)

III When you are finished with your work, you can disconnect by typing: `$ exit`

Your connection to the server may drop sometimes. In that case, you need to reconnect.

We advise you to watch the following video about the usage of SSH, which is used to connect remote servers, and SCP, which is used to transfer files between remote servers and your local machine: https://www.youtube.com/watch?v=rm6pewTcSro



```
[(base) simitii@Samets-MacBook-Pro ~ % echo "I am on my local machine now"
 I am on my local machine now
[(base) simitii@Samets-MacBook-Pro ~ % ssh sdemir20@linuxpool.ku.edu.tr
[sdemir20@linuxpool.ku.edu.tr's password:
 Last login: Sun Oct 18 14:30:40 2020 from 172.24.4.144
 -bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file o
 r directory
[[sdemir20@linux06 ~]$ pwd
 /Users/sdemir20
[[sdemir20@linux06 ~]$ echo "I am connected to the linuxpool now"
 I am connected to the linuxpool now
[[sdemir20@linux06 ~]$ exit
 logout
[(base) simitii@Samets-MacBook-Pro ~ % echo "I am back on my local machine again"
 I am back on my local machine again
 (base) simitii@Samets-MacBook-Pro ~ %
```

Figure 2: How to connect and disconnect using SSH

8. Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss the given assignments with your classmates, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the web material as everything on the web has been written by someone else. See Koç University - Student Code of Conduct.

9. Late Submission Policy

You may use up to 7 grace days (in total) over the course of the semester for the assignments. That is, you can submit your solutions without any penalty if you have free grace days left. Any additional unapproved late submission will be punished (1 day late: 20% off, 2 days late: 40% off) and **no submission after 2 days will be accepted.**