**INTERNATIONAL BURCH UNIVERSITY**

FACULTY OF ENGINEERING, NATURAL AND MEDICAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY



*Guessing game*

IBU 008 Programming I

*Aid Ajkunić*

Sarajevo, 2023

1. Introduction

BACKGROUND: The Guessing Game Program was initiated as an entertaining and engaging activity, designed to captivate users through the simple yet enjoyable act of guessing. Unlike traditional games that require extensive thinking or strategy, this program focuses on the immediate thrill of making accurate guesses. The development aimed to create a light-hearted and accessible experience, ensuring users of all ages could participate without the need for complex cognitive processes.

PURPOSE OF REPORT: This report is crafted to offer a detailed overview of the Guessing Game Program, emphasizing its simplicity and the joy it brings through the act of guessing. Readers can expect to gain insights into the program's development and its objectives.

SIGNIFICANCE OF THE PROGRAM: The Guessing Game Program holds significance as a stress-free and inclusive form of amusement. Its accessibility and lack of complexity make it an ideal source of entertainment for a diverse audience, contributing to a positive and enjoyable experience without the need for deep cognitive engagement.

ROADMAP:

1. INTRODUCTION:

-Briefly introduce the Guessing Game Program and its origins.

-Highlight the simplicity and accessibility of the program.

1. PROGRAM FUNCTIONALITIES:

-Explore the operational aspects of the Guessing Game.

-Examine how the program functions to provide an engaging experience for users.

1. FLOWCHART:

-Present a visual representation of the Guessing Game's flow.

-Break down the sequential processes through an illustrative flowchart.

1. CODE ANALYSIS:

-Delve into the underlying code of the Guessing Game Program.

-Provide insights into the technical aspects that drive the program.

1. RESULTS:

-Showcase the outcomes and impact of the Guessing Game.

-Highlight user experiences and the effectiveness of the program.

1. Program Functionalities
2. USER INTERACTION:

-Users are prompted with a menu at the beginning, offering options to play the game, view the leaderboard, or exit.

-The program ensures valid input from users, prompting them to enter a valid key if an incorrect input is provided.

1. GAMEPLAY:

-Upon choosing to play, users engage in a guessing game with multiple stages.

-The program randomly selects a starting letter, the number of letters in a word, and a specific word from the predefined database.

-Users guess the starting letter, the number of letters, and the actual word, with each stage affecting their score.

1. SCORING SYSTEM:
2. LETTER GUESSING:

-If users provide an incorrect letter guess, they incur a penalty of +1 point

1. NUMBER OF LETTERS GUESSING:

-An incorrect guess results in a penalty of +3 points added to the user's score.

1. WORD GUESSING:

-Each incorrect attempt incurs a penalty of +5 points to the user's overall score.

1. TOTAL SCORE:

-The user's final score is the sum of points accrued or deducted throughout the game.

-The objective is to achieve the smallest possible score, indicating a higher level of accuracy in guessing.

1. LEADERBOARD:

-Users can choose to view a leaderboard, which displays usernames and their respective scores.

-The leaderboard is stored in a file ("project\_leaderboard.txt") for persistent tracking of user performance.

1. FILE HANDLING:

-The program utilizes file handling to read and update the leaderboard.

-Usernames and scores are appended to the leaderboard file after each play session.

1. USER FEEDBACK:

-The program provides feedback to users at each stage, informing them of correct and incorrect guesses.

-Users are informed of their final score at the end of the game.

1. LOOP STRUCTURE:

-The program employs while loops to ensure continuous user engagement.

-Users can play the game multiple times or view the leaderboard without exiting the program.

1. GRACEFUL EXIT:

-Users have the option to exit the program, ensuring a user-friendly and graceful conclusion to the gaming session.

1. Flowcharts

In this report, flowcharts are employed as visual aids to illustrate key processes, sequences, and decision-making paths within the Guessing Game Program. Flowcharts serve as a valuable tool for elucidating the program's functionality, providing readers with a clear visual representation of the sequential and conditional aspects of various processes.

1. USER INTERACTION FLOWCHART:

This flowchart illustrates the user interaction process within the Guessing Game Program. It encompasses the menu options available to users, namely playing the game, viewing the leaderboard, or exiting the program.

Description:

-The process begins with the presentation of a menu to the user.

-The user's input is validated, ensuring it corresponds to a valid menu option.

-Depending on the selected option, the program directs the user to the respective section (play, leaderboard, or exit).

-The loop continues until the user chooses to exit the program.

1. GUESSING GAME WORKFLOW:

This flowchart provides an overview of the sequential steps involved in the main gameplay of the Guessing Game, covering letter guessing, number of letters guessing, and word guessing stages.

Description:

-The game starts with the random selection of a starting letter, the number of letters, and a specific word.

-Users proceed to guess the starting letter, with incorrect guesses incurring a penalty.

-The game then progresses to the number of letters guessing stage, where users guess the count of letters in the imagined word.

-Incorrect guesses lead to additional penalties.

-Finally, users attempt to guess the actual word, facing penalties for incorrect attempts.

-The game concludes by displaying the user's final score.

1. LEADERBOARD DISPLAY FLOWCHART:

This flowchart outlines the process of displaying the leaderboard, showcasing usernames and their respective scores.

Description:

-Users opt to view the leaderboard from the main menu.

-The program reads and retrieves data from the leaderboard file.

-The leaderboard is displayed, presenting usernames and scores in a readable format.

-Users are then prompted to return to the main menu or exit the program.

By incorporating these flowcharts, readers can gain a comprehensive understanding of the sequential processes and decision-making pathways within the Guessing Game Program. The visual representation enhances clarity and aids in the effective communication of the program's functionalities.

1. Code

import random          #importing module

alphabet="abcdefghijklmnopqrstuvwxyz"     #string with all letters in the alphabet

db={

    "a":[["aid","and","are"],["acid","ally","army"],["above","after","angle"],["absent","absurd","accept"]],

    "b":[["bad","bed","bar"],["bait","bank","belt"],["board","bread","break"],["backup","baking","ballet"]],

    "c":[["car","cat","cry"],["call","cake","cell"],["child","catch","cheap"],["cactus","camera","cancel"]],

    "d":[["day","dry","dog"],["dawn","dead","deal"],["dance","drink","daily"],["dagger","danger","decade"]],

    "e":[["ear","eat","end"],["easy","earn","east"],["empty","earth","enter"],["editor","effect","elixir"]],

    "f":[["far","fat","fan"],["fail","fake","fall"],["field","fault","fiber"],["falcon","family","father"]],

    "g":[["gas","get","gig"],["gate","gear","gene"],["gamer","glass","gloat"],["galaxy","gambit","garden"]],

    "h":[["hat","hog","hit"],["hair","head","half"],["habit","heavy","honey"],["hammer","hacker","handle"]],

    "i":[["ice","ick","ill"],["icon","idea","idol"],["idiot","issue","irony"],["ignore","incest","income"]],

    "j":[["joy","job","jun"],["jail","joke","join"],["jelly","juice","jewel"],["jersey","joking","jaguar"]],

    "k":[["key","kit","KIA"],["kiwi","kill","kick"],["kebab","knife","knock"],["kidnap","karate","keeper"]],            #mini database with words for every letter

    "l":[["lab","lap","leg"],["love","lake","long"],["labor","label","large"],["labour","layout","league"]],            #'x' and 'q' don't have words with 3 letters, so they have one less sublist from other letters

    "m":[["man","mom","mud"],["maid","make","more"],["magic","major","mango"],["magnet","makeup","manage"]],

    "n":[["new","net","now"],["name","nail","need"],["never","needy","niece"],["nectar","needle","nephew"]],

    "o":[["oil","owl","odd"],["omit","once","oven"],["ocean","occur","opera"],["occupy","office","obtain"]],

    "p":[["pet","pat","pay"],["pair","pace","pain"],["panda","panic","pants"],["palace","parent","parrot"]],

    "q":[["quiz","quad","quip"],["queen","quart","query"],["quotes","quests","quoted"]],

    "r":[["rat","raw","red"],["race","rain","rail"],["radar","reise","ratio"],["rabbit","racing","racist"]],

    "s":[["saw","sad","sun"],["soap","same","sale"],["salon","santa","scale"],["safari","safety","salary"]],

    "t":[["tag","tap","tax"],["tape","tail","tall"],["table","tango","teddy"],["tailor","talent","tablet"]],

    "u":[["use","USB","UNO"],["ugly","upon","user"],["uncle","union","until"],["unable","unfair","unique"]],

    "v":[["vet","van","var"],["vape","vein","vibe"],["value","valid","vault"],["vacuum","vaping","vector"]],

    "w":[["why","war","wet"],["what","warm","wall"],["waist","waste","water"],["window","wallet","warden"]],

    "x":[["xbox","xiao","Xray"],["xenia","xeric","xerox"],["xenial","xenias","xyloid"]],

    "y":[["yes","you","yet"],["year","yawn","yard"],["youth","young","yacht"],["yearly","yellow","yogurt"]],

    "z":[["zoo","zen","zap"],["zoom","zone","zero"],["zebra","zoned","zesty"],["zodiac","zombie","zipper"]],

}

The code begins by importing the random module. Subsequently, a string variable named 'alphabet' is declared, encompassing all the letters of the alphabet. Additionally, a dictionary named 'db' is established to serve as a mini-database, containing words associated with each letter.

command=""

print("Hello!")

command=input("Press 1 to play, press 2 to see a leaderboard, press 3 to exit!")        #starting menu which is making sure that user is pressing the right keys

while(command !="1" and command !="2" and command !="3"):

    command=input("Press valid key!")

The program begins with a simple menu system that greets the user and prompts them to choose an option by pressing '1' to play, '2' to see a leaderboard, or '3' to exit. The user's input is stored in the variable 'command.' To ensure valid input, a loop is implemented, allowing the user to re-enter their choice if an invalid key is pressed. This validation loop continues until the user provides a valid input, ensuring a smooth transition to their selected option.

while(command=="1" or command=="2"):

    if(command=="1"):

        print("Welcome to our little guessing game!")

        print("The goal is to have the smallest score!")          #asking user for his username

        username=input("Type your username!")

        print("For the first guessing task, you need to guess the first letter of a word I imagined!")

        print("For every miss you get +1 on your score!")

        first\_letter=random.randrange(0,len(alphabet))          #first\_letter is getting random number which represents a place of the letter in the alphabet

        score=0           #variable score represents the users score in the game

        letter\_guess=input("Your first guess is: ")

        while(letter\_guess not in alphabet):

            letter\_guess=input("Enter a letter, nothing else!")

        while(letter\_guess!=alphabet[first\_letter]):                         #asking user to guess the first letter of the word which is stored in the variable letter\_guess while making sure that the user is entering letter and nothing else

            letter\_guess=input("Wrong, try again! Your next guess is: ")     #if user doesn't guess right he gets +1 score, and program asks him to guess again

            while(letter\_guess not in alphabet):

                letter\_guess=input("Enter a letter, nothing else!")

            score+=1

        print("Good job!")

In this segment of the code, a loop is initiated as long as the user's input, stored in the variable 'command,' is either '1' or '2.' If the user selects '1,' the program welcomes the user to a guessing game and instructs them on the goal, which is to achieve the smallest score. The user is prompted to enter their username, stored in the 'username' variable.  
Subsequently, the program informs the user that the first task involves guessing the first letter of an imagined word. For each incorrect guess, the user's score, tracked by the 'score' variable, is increased by 1. The initial letter to be guessed is randomly chosen from the alphabet.  
The user is then prompted to make their first guess ('letter\_guess') and is provided with appropriate feedback. The program ensures that the user enters a letter and not something else, and continues the loop until some letter is written. If the user makes an incorrect guess, they receive feedback and are prompted to try again.

if(letter\_guess=="q" or letter\_guess=="x"):

            number\_of\_letter=random.randrange(0,3)               #computer randomly chooses how many letters is there going to be in the word and store the length of the word in the variable number\_of\_letter

        else:

                number\_of\_letter=random.randrange(0,4)

        print("Now try to guess number of letters of the word I imagined!")

        print("For every miss you get +3 on your score!")

        number\_guess=input("If your words starts with 'q' or 'x' try to guess a number between 4 and 6, or else try to guess number between 3 and 6. Your first guess of how many letters are there in the word I imagined is: ")

        while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

            number\_guess=input("Enter a valid number, nothing else!")

        while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):      #Protection from errors

            number\_guess=input("Enter a valid number, nothing else!")

In this part of the code, the program determines the number of letters in the word that the user needs to guess. If the randomly chosen first letter is 'q' or 'x,' the variable ‘number\_of\_letter’ is set to a random value between 0 and 2, as there are no three-letter words starting with 'q' or 'x.' For other letters, ‘number\_of\_letter’ is set to a random value between 0 and 3.  
The program then asks the user to guess the number of letters in the word. If the word starts with 'q' or 'x,' the user should guess a number between 4 and 6. Otherwise, they should guess a number between 3 and 6. The user's initial guess is stored in the variable ‘number\_guess’.  
The program includes a validation mechanism to ensure the user provides a valid number. If the first letter is 'q' or 'x,' the user is prompted to enter a valid number between 4 and 6, inclusive, and the loop continues until a valid input is received. For other letters, the user is prompted to enter a valid number between 3 and 6, inclusive. The loop ensures that the user provides a correct and valid number, preventing errors or alphabetic entries.

if(number\_of\_letter==0 and (letter\_guess=="q" or letter\_guess=="x")):

            while(number\_guess!="4"):                                           #checking if the user guessed the correct number, if not he gets +3 score, and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        if(number\_of\_letter==0 and letter\_guess!="q" and letter\_guess!="x"):

            while(number\_guess!="3"):                                          #checking if the user guessed the correct number, if not, he gets +3 score and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        if(number\_of\_letter==1 and (letter\_guess=="q" or letter\_guess=="x")):

            while(number\_guess!="5"):                                           #checking if the user guessed the correct number, if not he gets +3 score, and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        if(number\_of\_letter==1 and letter\_guess!="q" and letter\_guess!="x"):

            while(number\_guess!="4"):                                            #checking if the user guessed the correct number, if not he gets +3 score, and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        if(number\_of\_letter==2 and (letter\_guess=="q" or letter\_guess=="x")):

            while(number\_guess!="6"):                                            #checking if the user guessed the correct number, if not he gets +3 score, and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        if(number\_of\_letter==2 and letter\_guess!="q" and letter\_guess!="x"):

            while(number\_guess!="5"):                                           #checking if the user guessed the correct number, if not he gets +3 score, and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        if(number\_of\_letter==3 and letter\_guess!="q" and letter\_guess!="x"):

            while(number\_guess!="6"):                                           #checking if the user guessed the correct number, if not he gets +3 score, and he guesses again

                number\_guess=input("Wrong, try again! Your next guess is: ")

                while((letter\_guess=="x" or letter\_guess=="q") and (number\_guess.isalpha() or (number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                while(letter\_guess!="x" and letter\_guess!="q" and (number\_guess.isalpha() or (number\_guess!="3" and number\_guess!="4" and number\_guess!="5" and number\_guess!="6"))):

                    number\_guess=input("Enter a valid number, nothing else!")

                score+=3

        print("Good job! Now all you need to do is guess the word I imagined!")

In this section of the code, the program evaluates the user's guess for the number of letters in the word based on the randomly chosen first letter. If the first letter is 'q' or 'x,' there are only three possible options for the number of letters in the word since there are no three-letter words starting with 'q' or 'x.' For other letters, there are four options.

The program prompts the user to guess the number of letters in the word. The user receives a score penalty of +3 for each incorrect guess. The program ensures that the user enters a valid number, handling cases such as entering alphabetic characters or numbers outside the specified ranges. The user is prompted to enter a valid number until they provide a correct one.  
The code uses a series of if-else statements to check the number of letters and whether the first letter is 'q' or 'x.' It then enters a loop where the user is prompted to guess the number of letters, and their score is adjusted accordingly until they provide the correct answer.  
The last message in this section indicates that the user has successfully guessed the number of letters, and they are now ready to guess the entire word.

 print("For every miss you get +5 on your score!")

        print("Guess the word I imagined between these words: ", db[letter\_guess][number\_of\_letter])

        number\_of\_word=random.randrange(0,3)        #computer randomly chooses the word

        word\_guess=input("Your first guess is: ")

        while(word\_guess.lower()!=db[letter\_guess][number\_of\_letter][0].lower() and word\_guess.lower()!=db[letter\_guess][number\_of\_letter][1].lower() and word\_guess.lower()!=db[letter\_guess][number\_of\_letter][2].lower()):

            word\_guess=input("Enter the correct word!")                       #ensuring that the user correctly writes a word from the list

        while(word\_guess.lower()!=db[letter\_guess][number\_of\_letter][number\_of\_word].lower()):

            word\_guess=input("Wrong! Try again! Your next guess is: ")            #user guesses the word, if he doesn't pick correctly, he gets +5 on the score and guesses again

            while(word\_guess.lower()!=db[letter\_guess][number\_of\_letter][0].lower() and word\_guess.lower()!=db[letter\_guess][number\_of\_letter][1].lower() and word\_guess.lower()!=db[letter\_guess][number\_of\_letter][2].lower()):

                word\_guess=input("Enter the correct word!")

            score+=5

        print("Well done,",username,"your score is",score)

In this part of the code, the program informs the user that for every incorrect guess, they will receive a +5 score penalty. The program then displays the possible words for the user to guess based on the previously determined letter and number of letters in the word.  
The computer randomly selects one of these words, and the user is prompted to make their first guess. The program ensures that the user correctly inputs a word from the provided list. If the user's initial guess is incorrect, they receive a +5 score penalty, and the program prompts them to guess again until they provide the correct word.  
The final message indicates the end of the game, displaying the user's username and their score.  
If the user inputs a word that is not in the list of available words, the program will notify the user that their input is incorrect. Specifically, the user will receive a prompt asking them to try again and enter a word that is part of the provided list. The program will not proceed until the user provides a valid input, i.e., a word from the list.  
This mechanism is in place to ensure that the user follows the rules of the guessing game and selects a word that the program has predefined. Each time the user enters an incorrect word, they will be prompted to make another attempt until they successfully input a word from the list.

f=open("project\_leaderboard.txt","a")

        f.write(username+" "+str(score)+"\n")       #storing users score in the file called project\_leaderboard.txt

        f.close()

        command=input("Press 1 to play, press 2 to see a leaderboard, press 3 to exit!")

        while(command !="1" and command !="2" and command !="3"):                       #asking user what does he want to do now

            command=input("Press valid key!")

In this part of the code, the program opens a file named "project\_leaderboard.txt" in append mode ("a") to store the user's score and username. The user's information, including their username and score, is written to the file with a newline character (\n) to separate entries. Subsequently, the file is closed to save the changes.  
After updating the leaderboard file, the program prompts the user with a menu, allowing them to choose the next action. The user can press: "1" to play the guessing game again, "2" to view the leaderboard, "3" to exit the program.  
If the user enters an invalid key, they will be prompted to press a valid key until they provide a correct input.

if(command=="2"):

        f=open("project\_leaderboard.txt","r")

        for line in f:                          #letting user see the list from the file project\_leaderboard.txt

            print(line)

        f.close()

        print("Press 1 to play, press 2 to see a leaderboard, press 3 to exit!")    #asking user what does he want to do now

        command=input()

        while(command !="1" and command !="2" and command !="3"):

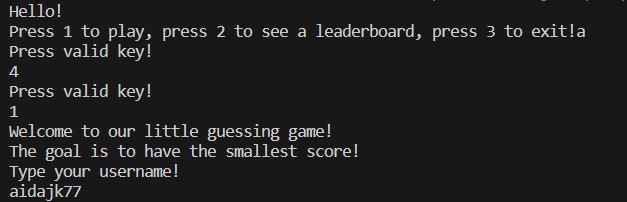
            command=input("Press valid key!")

print("Goodbye")

In this section of the code, if the user inputs "2," the program opens the "project\_leaderboard.txt" file in read mode ("r") to display the contents of the leaderboard. It then iterates through each line in the file and prints the leaderboard entries.  
After displaying the leaderboard, the program displays the starting menu to the user.  
The program will print "Goodbye" and exit when the user chooses to exit the program by pressing "3."

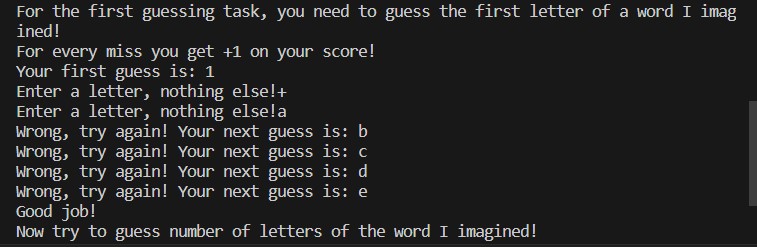
1. Results

* **Starting menu and user registration process**



In this screenshot, the user is prompted to interact with the program by entering a numeric choice. Initially, an invalid input (4) is entered, and the program requests a valid key. Upon entering '1', the user is welcomed to the guessing game. The game's objective, which is to achieve the smallest score, is explained, and the user is prompted to enter their username.  
This screenshot showcases the user registration process. The player, with the username 'aidajk77', has successfully entered their name. The registration step is a crucial part of personalizing the gaming experience and allows tracking of individual scores. Once registered, the player can proceed to engage in the guessing game.

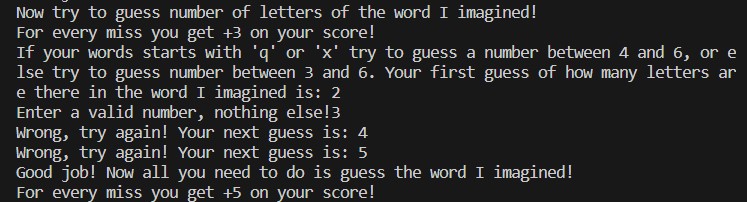
* **First guessing task – Letter guessing**



In this screenshot, the player has entered the guessing phase where they need to identify the first letter of the imagined word. The initial guess ('1') is incorrect, and the program prompts the player to enter a letter only. The player inputs 'a', but it's also incorrect, leading to subsequent attempts with 'b', 'c', 'd', and 'e'. Finally, the correct guess of the letter 'e' is made, and the program acknowledges the correct response with a "Good job!" message.

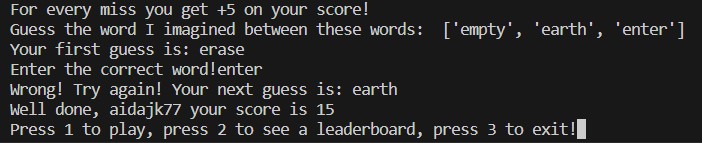
This screenshot illustrates the score tracking and progress after the first guessing task. Each incorrect guess increments the player's score. In this example, the player had four incorrect attempts ('a', 'b', 'c', 'd'), resulting in a score of 4. The player is now set to proceed to the next phase of the game with the objective of minimizing their score.

* **Second guessing task – Number of letters**



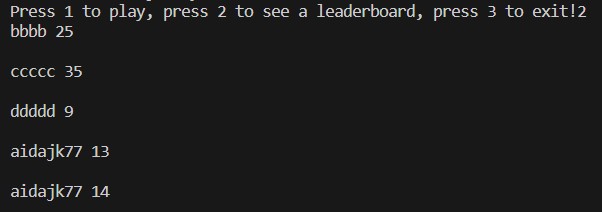
In this screenshot, the player has transitioned to the second guessing task, where they need to determine the number of letters in the word imagined by the program. The initial guess of '2' is incorrect, and the program prompts the player to enter a valid number. The player provides '3', which is also incorrect, leading to subsequent attempts with '4' and '5'. The correct guess of '5' is made, and the program acknowledges this with a "Good job!" message. In this example, score got bigger by +6.

* **Final challenge – Guess the word**



In this screenshot, the player has three attempts to guess the word among the provided options: ['empty', 'earth', 'enter']. The initial guess of 'erase' is incorrect, and the program prompts the player to enter the correct word. The second guess of 'enter' is also incorrect, and the program again prompts the player for the correct word, emphasizing that guesses outside the given options do not affect the score. The third guess of 'earth' is correct, and the program acknowledges this with a "Well done!" message. The player's final score of 15 is displayed, which includes penalties for incorrect guesses in both the letter and number guessing tasks.

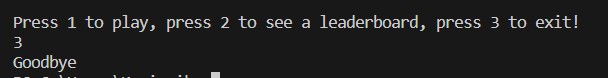
* **Leaderboard display**



In this leaderboard display, the program presents the top scores achieved by different players. Each entry consists of a username and their corresponding score.

It allows players to gauge their performance relative to others and adds a competitive element to the gaming experience.

* **Exit message**



In this final screenshot, the player has chosen to exit the game by pressing '3'. The program responds with a simple and friendly "Goodbye" message, indicating that the game session has ended. This screen serves as a concluding message, providing closure to the user's interaction with the program.