|  |  |  |  |
| --- | --- | --- | --- |
| **Coursework #1** | | | |
| **Module’s Information:** | | | |
| Module | | ICT2102 Introduction to Data Structure | |
| Session | | January 2020 | |
| Programme | | DITN | |
| Lecturer | | **RAJERMANI A/P THINAKARAN** | |
| Email: rajermani.thina@newinti.edu.my  Phone: 06-7982000 - 2209 | Room: A3-15 |
| Coursework Type | | Assignment (Max 3 members/group) | |
| Percentage | | 20% out of 100% | |
| Hand-out Date | | Week 5 (18/2/2020) | |
| Due Date | | Week 8 (10/3/2020 Before 5pm) | |
| **Students’ Declaration:** | | | |
| ***We declare that:***   1. ***We understand what is meant by plagiarism*** 2. ***This assignment is all our own work and we have acknowledged any use of the published or unpublished works of other people.*** 3. ***We hold a copy of this assignment which we can produce if the original is lost or damaged.***   **[Name/ID] \_\_\_\_\_NG IRIS / I18015132\_\_\_\_\_\_\_\_\_\_\_\_ [Signature] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **[Name/ID] \_\_\_\_\_ONG XING YUE / I18015044\_\_\_\_\_ [Signature] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **[Name/ID] \_\_\_\_\_TEO YI TENG / I18015272­­\_\_\_\_\_\_\_ [Signature] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **[Date] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | |
| **Learning Outcomes Assessed:** | | | |
| 1 | Develop and compare various types of searching and sorting algorithms using basic algorithm analysis tools. | | |
| 2 | Read and design recursive algorithms. | | |
| 3 | Describe and apply appropriate data structures concepts. | | |
| 4 | Design and apply appropriate data structures to develop applications. | | |

|  |
| --- |
| **Description of Coursework #1:** |

For this assignment, you will be making a simple version of the [Hangman](http://en.wikipedia.org/wiki/Hangman_(game)) Word Guessing game with. This game will be played in the terminal window. In this game, the computer picks a random word from a pre-supplied list (from a text file). The rules are same as the classic hangman game which are

1. The player tries to solve to puzzle by guessing one letter at a time.
2. Every time a guess is correct, all the characters in the word that match the guess will be ``turned over.'' For example, if your guess is ``o'' and the word is ``book'', then both ``o''s in the solution will be counted as ``solved.''
3. Every time a wrong guess is made, a stroke will be added to the drawing of a hangman, which needs 7 strokes to complete (Figure 1). Each unique wrong guess only counts against the player once.
4. If the drawing of the hangman is completed before the player has successfully guessed all the characters of the word, the player loses.
5. If the player has guessed all the characters of the word before the drawing is complete, the player wins the game.
6. If the player does not guess enough letters to either win or lose, then display appropriate message.

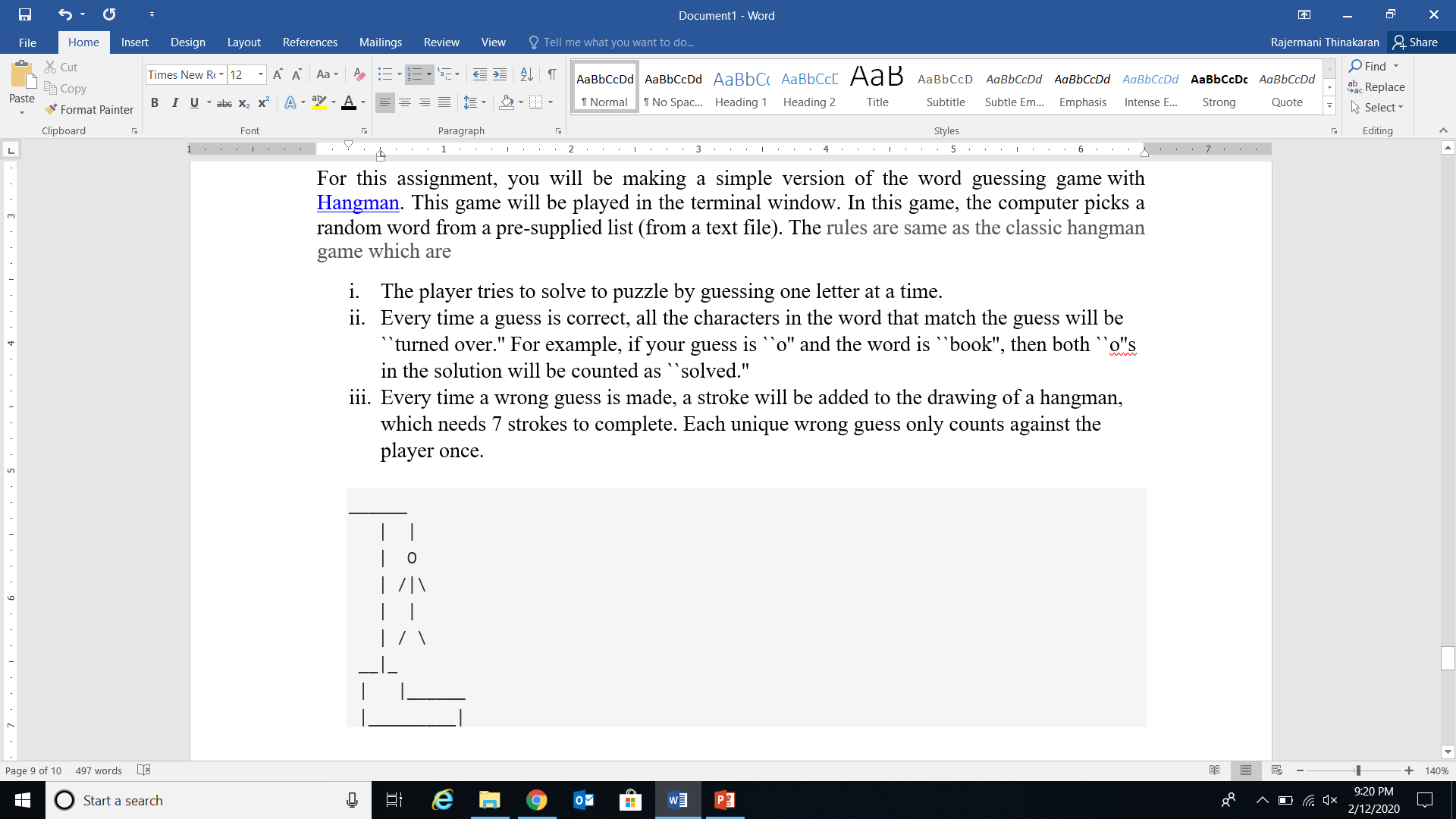


Figure 1

On top of that, your game also need to keep track of previously guessed letters and tell the user. The application also able to record and retrieve the players’ game history’s such as how many time the player win, lose or draw; how many attempts the player won the game or any other information. Make sure your game runs until the player decides to quit.

To develop the Hangman Word Guessing game, the following algorithms must be used.

1. Searching algorithm - Linear / Sequential search, Binary Search. Either one or both.
2. Sorting algorithm – Selection sort, Bubble sort, Insertion sort, Merge sort or combination of this algorithms.
3. Recursion
4. Hash Function and Collision Resolution

Notes:

* You should have at least five functions, excluding main function.
* Do not use the any built-in library for the above listed algorithms.

**Submit your report with the following details: -**

1. Introduction
2. Algorithm (Pseudocode or Flowchart)
3. Details of the testing process (test cases and test data)
4. Screen Output
5. Summary
6. Attached the Rubrics (Group and Individual)
7. Include the source code as appendix and upload in the BB.

Each group are required to do 20 minutes presentation with the PowerPoint and demo the developed game.

|  |
| --- |
| **End of Coursework #1** |

**Table of Contents**

[Introduction 5](#_Toc34691248)

[Algorithm 6](#_Toc34691249)

[main() function 6](#_Toc34691250)

[menu() function 6](#_Toc34691251)

[enterName() function 6](#_Toc34691252)

[drawHangman() function 7](#_Toc34691253)

[printWordAndResult() function 7](#_Toc34691254)

[checkWord() function 7](#_Toc34691255)

[printMessage() function 8](#_Toc34691256)

[startGame() function 8](#_Toc34691257)

[validateInput () function 10](#_Toc34691258)

[printLife() function 10](#_Toc34691259)

[quitGame() function 10](#_Toc34691260)

[loadRandomWord() function 10](#_Toc34691261)

[printAvailableLetters() function 10](#_Toc34691262)

[triesLeft() function 11](#_Toc34691263)

[printLetters() function 11](#_Toc34691264)

[struct leader 11](#_Toc34691265)

[class hash 11](#_Toc34691266)

[Hash() function 12](#_Toc34691267)

[PrintTable() function 12](#_Toc34691268)

[Unit Test 13](#_Toc34691269)

[Integration Test 15](#_Toc34691270)

[Screen Output 16](#_Toc34691271)

[Summary 17](#_Toc34691272)

[References 18](#_Toc34691273)

# Introduction

In this assignment, there will be profoundly present the method of create a basic form of the Hangman Word Guessing game. Hangman Word Guessing game is speculating word amusement with a hanging man to record the chances of speculating the letter. The concept of the diversion will be the system haphazardly select the word from the data recorded and the underlined letters will show up within the word whereas player speculated redress something else the draw of the man hanging will be prepare one step to wrap up the diversion in our diversion will be 7 chances for player to guess wrongly. The game of hangman was conducive to moved forward player’s ability and knowledge. Learning process incorporate game can be fun and effectively to have the impression and progress the productivity of learning. This game can be play with multiplayer which can make strides the social ability. (Prastuti, et al., 2013)

The Hangman Word Guessing game that will be clarify approximately the algorithm, details of the testing process with integrate, unit and overall system test and screen of output. We are going to develop and compare various sorts of searching and sorting algorithms, update recursive calculations and give depiction around the data structures concepts to create the Hangman Word Guessing game. The theme of our develop game is country name, it will have 40 countries name arbitrarily select one of the country for user to guess. Either user lose or win the game， user have a choice to proceed or exit the game, the score will be record each user guess when user is rectify guessed, and the score will bring to next game until user exit the game. When the score of user is hit the score within the leader board, user’s name will be appear within the leader board in the menu.

In a nutshell, the Hangman Word Guessing game we developed will utilizing C++ language with searching, sorting, recursion and hash function and collision resolution calculation. Those algorithms that we develop the Hangman Word Guessing game will expand by provide algorithm, testing and output screen in this assignment.

# Algorithm

The details of the code for develop the Hangman Word Guessing game will appear with depiction concurring to the function incorporate within the game.

## main() function

1. start with welcome screen

2. go to menu

3. pass the global variable that hold playerName and score to hash table

## menu() function

1. prompt user to enter choice

1. Start game
2. View leaderboard
3. Exit the program

2. while (choice > 3 or choice < 1)

prompt user to enter their choice again

end while

3. if (choice == 1)

if (playerName variable is not empty)

go to changePlayer() to check if the player want to start with a new player name

else

go to enterName() to prompt player name

end if

else if (choice == 2)

go to leaderboard() to check their ranking

else if (choice == 3)

exit the program

end if

## enterName() function

1. prompt player to enter player name after playing game.

2. validate player name. While player name is empty, prompt again

3. If a valid input call the Hash() function to calculate index to store playerName and score.

## drawHangman() function

1. if (wrong guessed >= 1)

draw “|”

end if

if (wrong guessed >= 2)

draw “|”

end if

if (wrong guessed >= 3)

draw “o”

end if

if (wrong guessed >= 4)

draw “/|\”

end if

if (wrong guessed >= 5)

draw “|”

end if

if (wrong guessed == 6)

draw “/”

end if

if (wrong guessed >= 7)

draw “/ \”

end if

## printWordAndResult() function

1. pretend won = true

2. go to checkWord() to get the string which used to update the blanks

3. for (range between the length of returned string)

If (found ‘\_’ in the returned string)

won = false (means the string is not completed)

end if

end if

4. print returned string to update the blanks

5. return won variable to startGame()

## checkWord() function

1. pretend found = false

2. if (every letter of the word is checked)

return string which used to update the blanks to printWordAndResult()

end if

3. for (range between the length of guessed string)

If (one letter in guessed string matched one letter in the word)

found = true (means the guessed letter is found in the word)

add the matched letter into empty string

end if

end for

If (found == false)

add blank “\_” into empty string

end if

4. return to the same function to check next letter in the word

## printMessage() function

1. if (printTop)

print top line

print left line

else

print left line

end if

2. pretend front = true

for (range between message length and 33) -> to print message in the center

If (front)

message += “ “ + message;

else

message = message + “ “;

end if

front = false

end for

print message

3. if (printBottom)

print right line

print bottom line

else

print right line

end if

## startGame() function

1. pretend life = 7, tries = 0, win = false

2. get random word from loadRandomWord()

3. do

print interface

get game status (win) from printWordAndResult

if (win)

break

end if

if (tries == 7)

break

end if

get guessed letter from player

if (guessed letter == 1)

check if player really wants to quit the game [Y/N]

if (confirm == ‘Y’)

go to menu()

else

print message to continue the game

prompt letter from player

end if

end if

do

found = false

if (guessed letter is non-alphabet and not equal to 1)

found = true

print invalid message

prompt letter from player

end if

for (range between the length of guessed string)

if (player input matched letter in guessed string)

found = true

print message the letter is taken before

prompt letter from player

end if

end for

end do while (found == true)

guessed string += player input

go to triesLeft() to get the number of failed attempts (tries)

life -= tries

end do while (tries <= 7)

if (win)

print won message

score += life

else

print game over message

print the actual word

end if

check if player wants to play again (playAgain)

if (playAgain == ‘Y’)

go to startGame()

else if (playAgain == ‘N’)

go to menu()

end if

## validateInput () function

1. Get user’s input from changePlayer() or startGame() or leaderboard()
2. Pretend input to uppercase.
3. while (input != 'Y' && input != 'N')

print message Invalid input. Please enter again.

prompt input from player

end while

return input

## printLife() function

1. Get heart from startGame()
2. while(heart>0)

print message \*

heart- -

end while

## quitGame() function

1. Function will be call from menu()
2. Print message Thank you for your support.

## loadRandomWord() function

1. loadRandomWord() will be called by startGame()
2. pretend count=0
3. pretend countryName[] load 40 country
4. Load countryName[] into file “countryName.txt”
5. Random select country from “countryName.txt”
6. while (getline(counrtyRead, word))

count++

if(count==random)

return word

endif

end while

## printAvailableLetters() function

1. Get values from startGame()
2. 26 characters load into 2 array alphabets1[13] and alphabets2[13]
3. Call printMessage() with parameter “AVAILABLE LETTERS”
4. Call printLetters() with the parameter of the values from startGame() and alphabets1
5. Call printLetters() with the parameter of the values from startGame() and alphabets2

## triesLeft() function

1. Get values from startGame() with the parameter of wordToGuess and guesses
2. Pretend error=0
3. for (int i = 0; i < guesses.length(); i++)

if (wordToGuess.find(guesses[i]) == string::npos)

error++;

end for

return error

## printLetters() function

1. Values get from printAvailableLetters() with the parameter of taken and alphabets
2. Prentend found = false
3. Pretend letters = “ “
4. for (char i = 0; i < 13; i++)

for (char j = 0; j < taken.length(); j++)

if (alphabets[i] == taken[j])

found = true;

End if

End for

if (!found)

letters += alphabets[i];

letters += " ";

End if

found = false;

End for

1. call printMessage()

## struct leader

{

string name;

int score;

leader\* next;

};

## class hash

1. Create a constructor in hash class for hash class that initialize each value in hash table as default.
2. Create a AddItem() function in the class.

private:

leader\* HashTable[tableSize];

public:

hash()

for (int i = 0; i<tableSize; i++)

set the default value for the variable in struct.

AddItem() function

create file as ofstream which doesn’t overwrite the data.

Hold value as int index that return by Hash() function

Create pointer to hold the value.

if (HashTable[index]->name == "empty")

overwrite value into hash table which are empty

open file and write the value into text file

else

pointer + 1 and store the value to the index

while(Ptr->next != NULL)

to loop and found last pointer and add into the index that without element

Set pointer Ptr->next = n;

## Hash() function

1. Pass the global variable playerName.
2. Initialize two variable which hash = 0 and index as integer.
3. for (int i = 0; I < key.length(); i++)  
    hash = hash + (int) key [i];

index = hash % tableSize;

return index;

## PrintTable() function

1. open a file to get the value that store in text file.
2. Initialized a variable to open file.
3. Initialized two variables which textname as string and textscore as int.

while((the file did open and read)&& count < tableSize)

if(textname != "empty")

cout the data as textname and textscore

++count;

Cout the playerName and score here.

## Struct unsortArray

declare string unSortPlyName;

declare int unSortScore;

## void syncSortLeader( )

while((print>>textname>>textscore)&& count < tableSize)

if(textname != "empty")

if (count < 10)

Stored in the global struct

count++;

Call decSortArray( );

## 

## void decSortArray( )

for(int step = 1; step < 10; step++)

declare array to store the struct information.

int j =step - 1;

while(sortScore > firstArr[j].unSortScore && j >= 0)

store the data in descending order.

Pretend firstArr[j+1].unSortScore=sortScore;

Pretend firstArr[j+1].unSortPlyName=sortPlyName;

## Unit Test

Unit testing focused on each individual unit in the system. Unit test is implemented to validate that each unit in the system is able to perform as expected.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Unit test | Situation | Expected Result | Result  (True /False) |
| 1. | Program start | Player start running the program. | System will display a welcome screen. | True |
| Player press any key on the welcome screen to continue. | System will show a menu after Player press any key on the welcome screen. | True |
| 2. | Welcome screen | Player start running the program. | System will display a welcome screen. | True |
| 3. | Menu | Player press any key on the welcome screen. | System will display a menu and prompt player to enter their choice. | True |
| Player enter integer > 3 or < 1. | System will display invalid message and prompt player to enter again. | True |
| Player enter integer 1. | Start the game. | True |
| Player enter integer 2. | System will show leaderboard. | True |
| Player enter integer 3. | System will display thank you message and quit the program. | True |
| 4. | Hangman drawing | Player enter a wrong letter. | System will show “|” as the player failed 1 time. | True |
| Player enter 2 wrong letters. | System will show “|” as the player failed 2 time. | True |
| Player enter 3 wrong letters. | System will show “o” as player user failed 3 time. | True |
| Player enter 4 wrong letters. | System will show “/|\” as player user failed 4 time. | True |
| Player enter 5 wrong letters. | System will show “|” as the player failed 5 time. | True |
| Player enter 6 wrong letters. | System will show “/” as the player failed 6 time. | True |
| Player enter 7 wrong letters. | System will show “/ \” as the player failed 7 time. | True |
| 5. | Hangman game interface | Player enter ‘1’ in the menu to start the game. | System will show the game interface with frames and display the message inside the frames neatly. | True |
| 6. | Word checking | Player enter a letter to guess the word. | System will store the correct letter in its position and store an underline if the letter is incorrect. | True |
| 7. | String printing | String returned from word checking function. | System will update the string with blanks and correct guessed letter to inform the player whether he guessed the letter correctly. | True |
| 8. | Player name | Player decided to quit the game. | System will prompt the player to enter a name. | True |
| 9. | Word guessing | System want to display the number of letters in the word for player to guess. | System will display few underlines which is same with the number of the actual word. | True |
| Player enter a letter. | The letter which is taken will disappeared from the list of available letter. | True |
| Player used up 7 chances. | The system will display the answer and message which shows game over.  System will ask player if they want to play again. | True |
| Player guessed the word correctly. | System will display win message and ask player if they want to play again. | True |
| Player enter [Y/y] to confirm they want to play again. | System will start a new game. | True |
| Player enter [N/n] to refuse playing again. | Player will be directed to the menu page. | True |
| Player enter invalid input when system asked if they want to play again. | System will show invalid message and prompt the input again. | True |
| 10. | Player’s life | Player enter a letter to guess the word. | The number of life will be updated and shown according to the number of chances left. | True |
| Player guessed a wrong letter. | System will deduct the life. | True |
| Player start a new game. | System will reset the number of life to 7. | True |
| 11. | Score record | Player confirm to start a new game with same player name. | The score earning from previous game will be shown correctly and the number will continue to increment with the current score. | True |
| 12. | Store Name and Score | Player decide to quit game [y/Y] to menu page or [N/n] directly quit the system. | The name and score will be store in the hash table. After that, store in a text file without overwrite the preview record. | True |
| System will display player name and score in leader when player go to leaderboard from menu. | System will display the preview record from text file to leaderboard and current player name and score. | Ture |
| 13. | Print the leaderboard | Player choose 2 in the menu. | System will display the leaderboard as descending score. | Ture |
| Player back to menu and choose 2 to leaderboard. | System will display the current player name and score | Ture |
| 14. | Quit game | Player enter ‘1’ to quit the game. | System will print out confirmation message to confirm player action. | True |
| Player enter [Y/y] to confirm their action. | System will return to menu page. | True |
| Player enter [N/n] to confirm their action. | System will show message to continue the game. | True |
| Player enter invalid input. | System will show invalid message and prompt player to enter again. | True |

## 

## Integration Test

Integration test is implemented to test the interaction between groups of units.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Integration test | Situation | Expected Result | Result  (True /False) |
| 1. | Update game status | Player guessed a wrong letter. | Number of strokes increased until the hangman drawing is completed. System will deduct the number of life and update life status to the player. | True |
| 2. | Update available letters | Player entered a guessed letter. | The taken letter will disappeared from the list of available letters. | True |
| Player guessed the word by entering more than one letters. | The correct letters will be updated to the string and incorrect letters will cause deduction of chances. | True |
| Player entered a letter which is not in the list of available letters | System will display proper message and prompt player to enter again. | True |
|  | Update guessed string | Player entered a letter to guess the word. | The correct letter guessed will appears on the empty underline in the right place. If multiple underlined have the same character will appear too. | True |
|  | Update name and score | Player enter a player name | The system will accept the player name which valid and store in text file. | True |
| Player score after playing game | The system will store the score of current players with player name into a text file | True |
|  | Show game result | Player guessed the word correctly. | System will display win message and ask if the player wants to play again. | True |
| Player used up 7 chances. | System will display correct answer and game over message. Then ask if the player wants to play again. | True |

# Screen Output

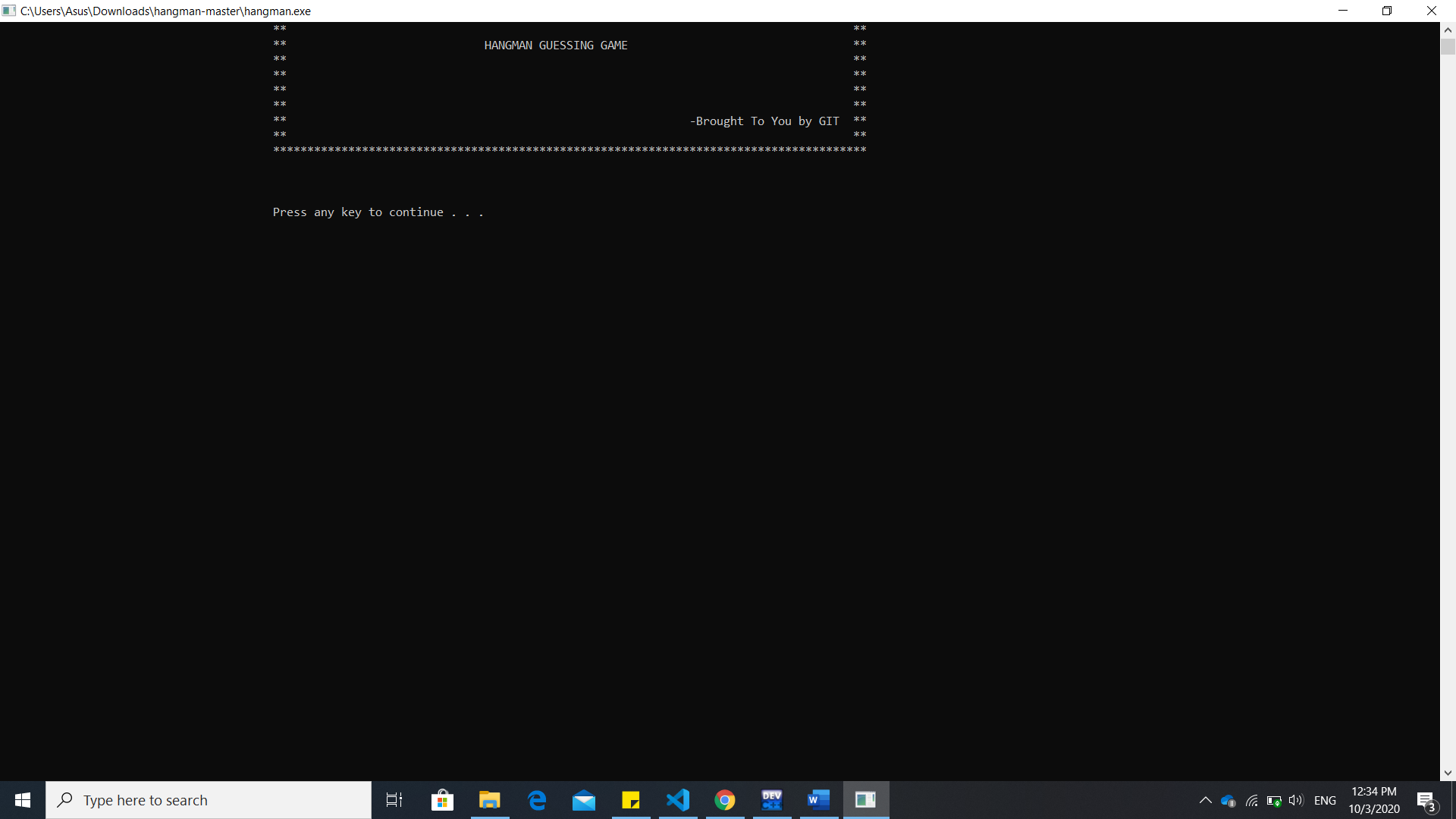


Figure 1: The homepage of hangman game where the user will be direct to the system.

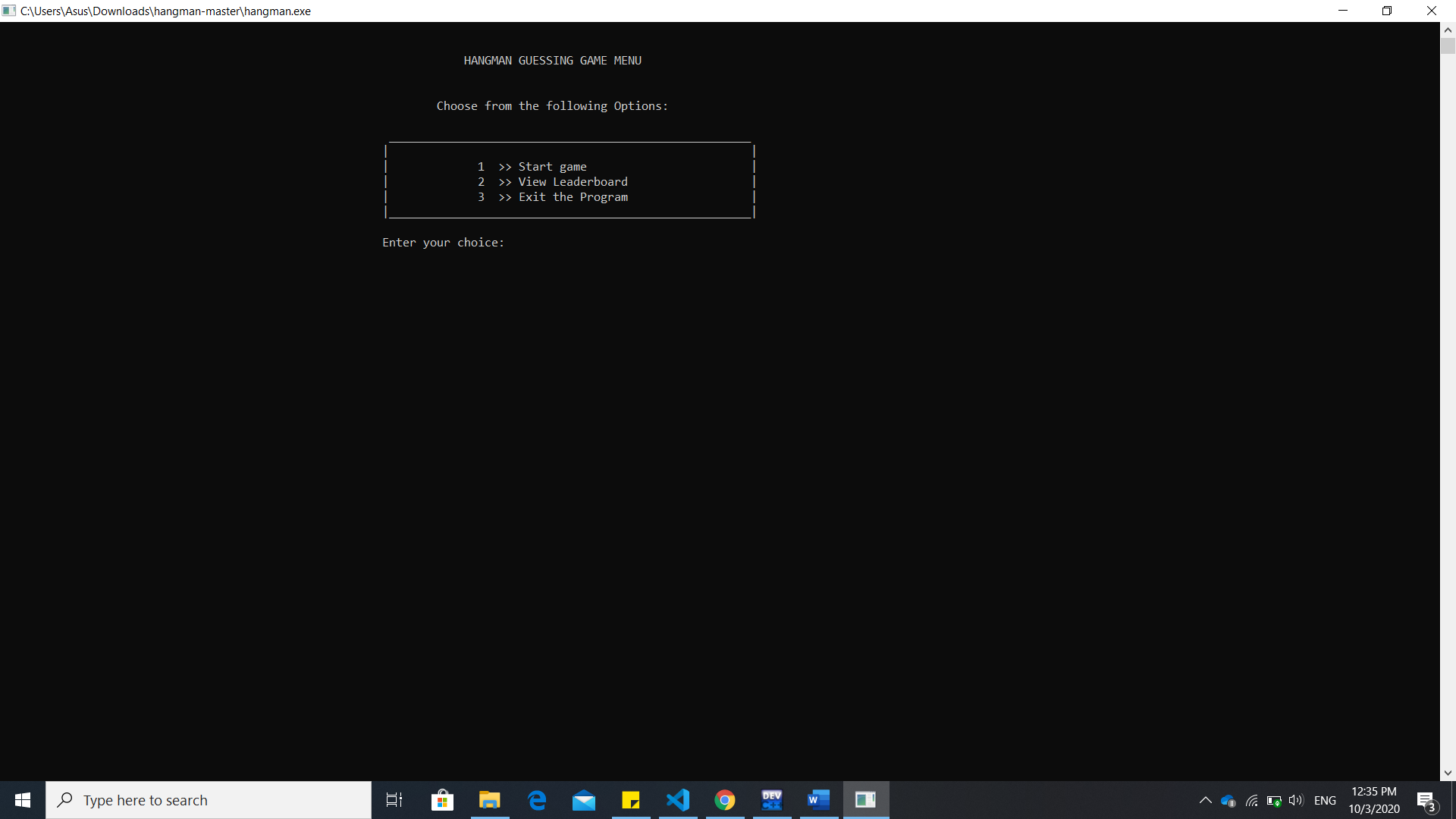


Figure 2: The menu page after the user press any key in homepage.

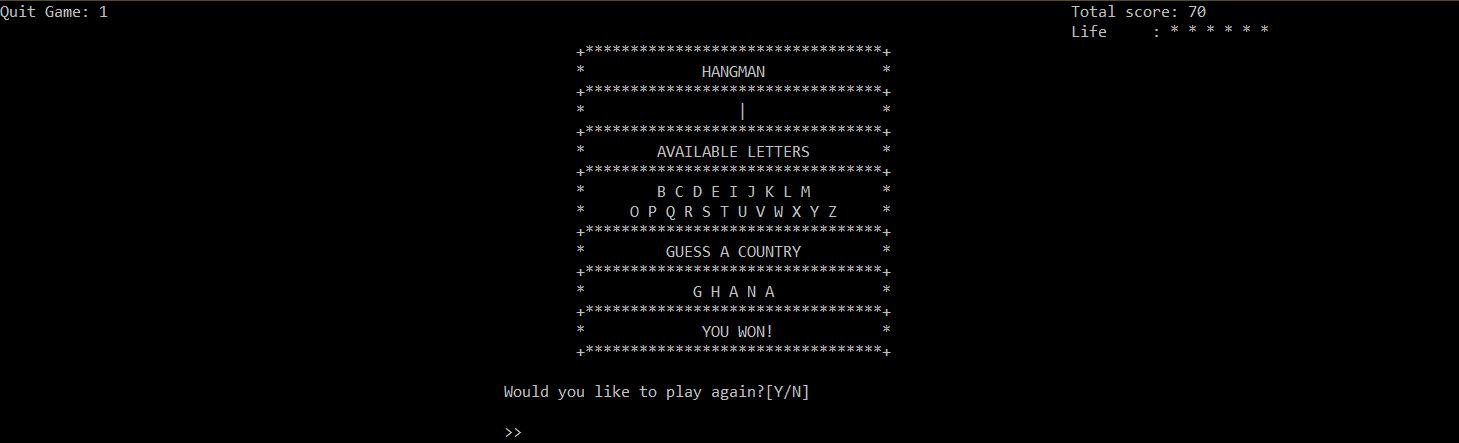


Figure 3: When user press 1 in menu page the system will direct user to this interface and start playing the game.

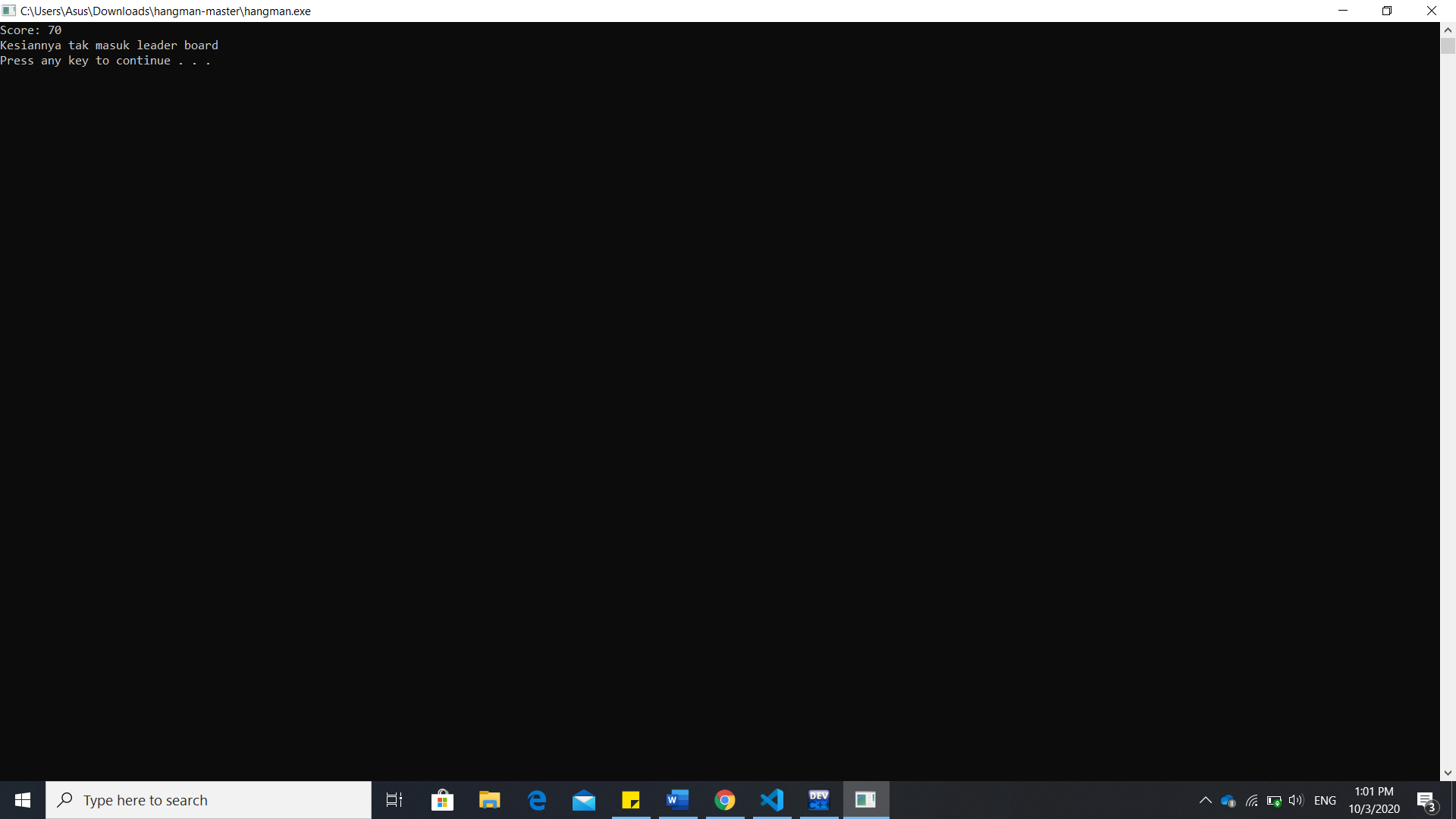


Figure 4: This massege will be display to user who didn’t hit the leaderboard after playing the hangman game.

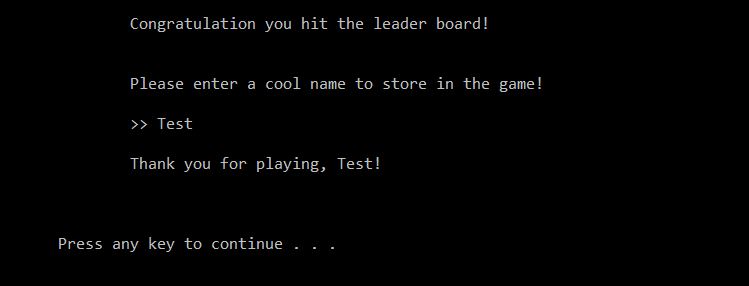


Figure 5: If the user hit the score, the system will request user to enter the name.

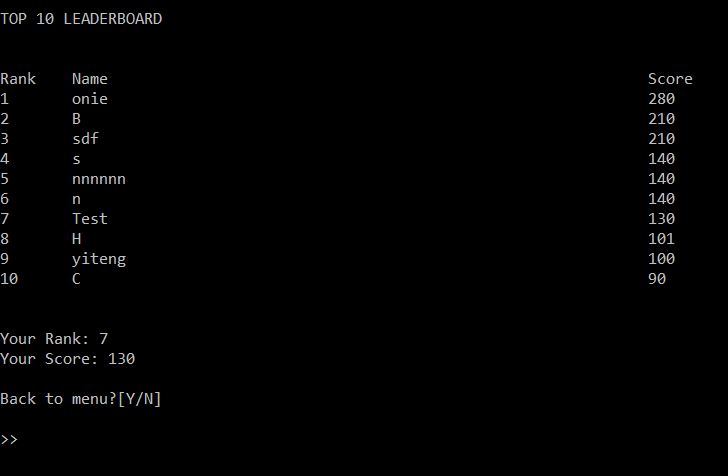


Figure 6: After playing the game, the user can view the leader board and if they hit the score the system will display their ranking and score.

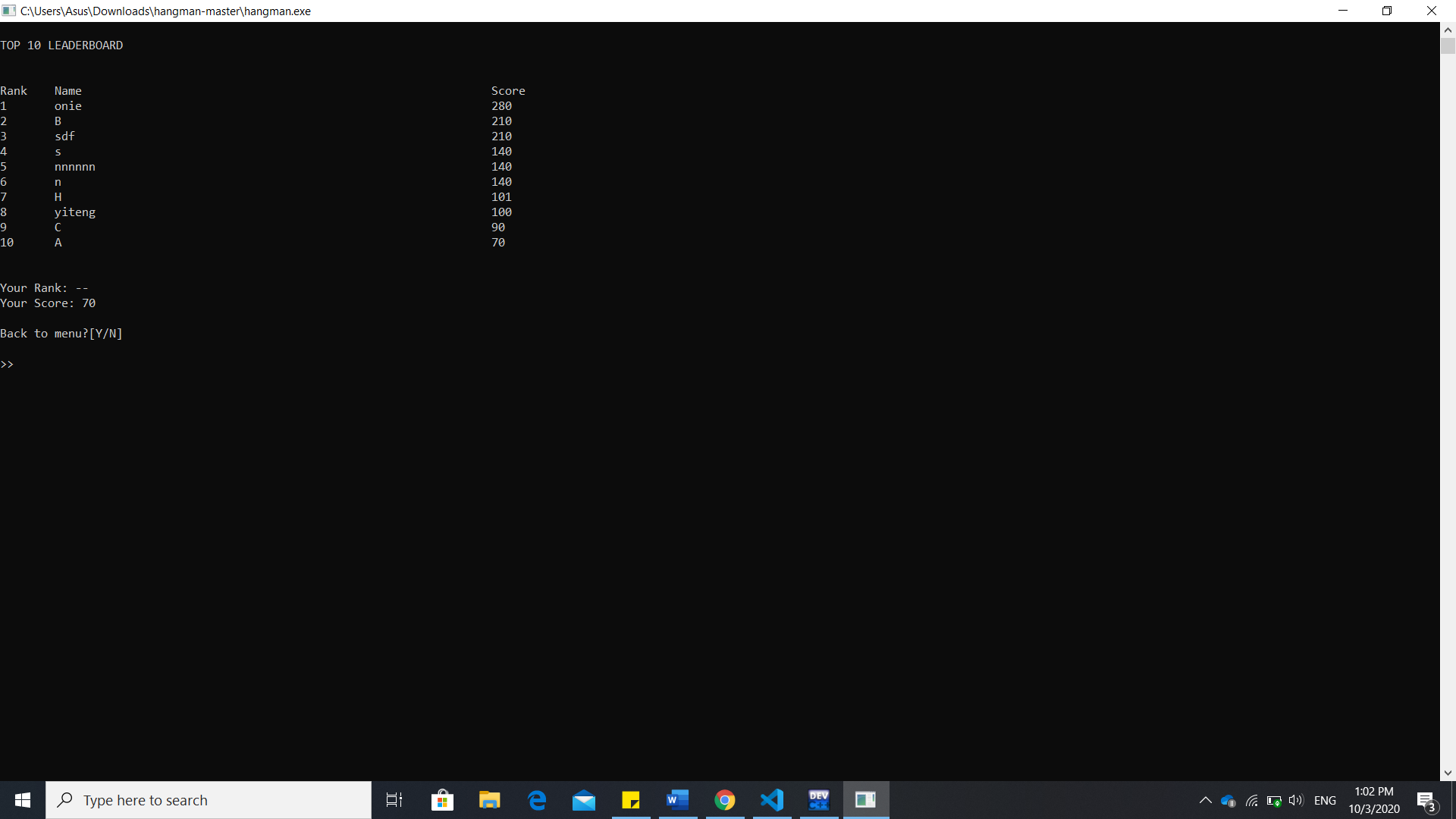


Figure 7: The user who press 2 in the menu page will be direct to leaderboard

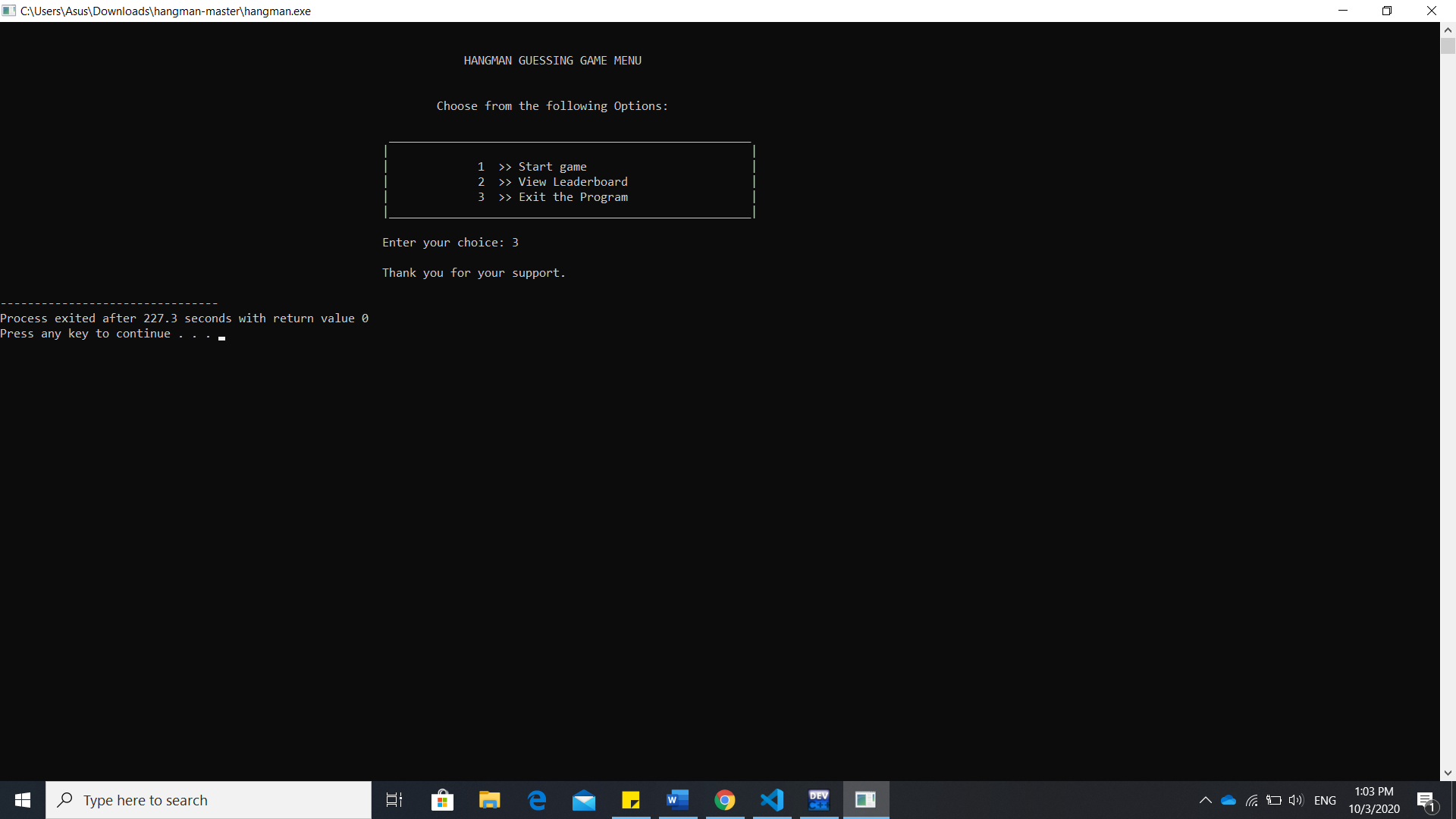


Figure 8: The message will be prompt out when user press 3 in menu page. Summary

In conclusion, the hangman project can be considered one of the fun projects that have been done. It is because it not only a group project but it also a game that need to be create with the creativity. The project did specific the algorithm that need to include in the development. And it has been created by referring to the internet and the lecturer note that given by the lecturer. In the beginning, to developing this project a research has been carry to each algorithm and explore the ways to use its. After gather the information, the project has been developed with the help of information. While develop the system, there are a lot of extra knowledge about C++ have been learnt. Besides, the deep understanding of each algorithm has been carried out after developing this project. At the end, our group did enjoy developing the system while playing it. At here, we wish to thank to our lecturer miss Rajermani who always help when we need. And thanks to each of the group member who

# References

Prastuti, N., Salam, U. & Novita, D., 2013. Improving Students Ability In Asking And Giving Opinion By Using Hangman Game. *Jurnal Pendidikan dan Pembelajaran U*

**ICT2102 Assignment 1 Rubrics**

**Group Component**

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Group Marks: \_\_\_\_\_\_/ 50**

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Performance** | | | | | **Marks Given** |
| **Poor (0 – 2 marks)** | **Fair (3 – 4 marks)** | **Good (5 – 6 marks)** | **Very Good (7 – 8 marks)** | **Excellent (9 – 10 marks)** |
| **Understands the Problem and Requirements** | Student’s work shows incomplete understanding of problem and/or requirements. | Student’s work shows slight understanding of problem and requirements. | Student’s work shows understanding of problem and most requirements. | Student’s work shows complete understanding of problem and all requirements. | Student recognizes potential conflicts b/t requirements and seeks clarification from client/user. |  |
| **Uses Appropriate Algorithms** | Student ‘hacks out’ program with no thought to algorithm design. | Student chooses/ designs algorithm(s) that are incorrect. | Student chooses/ designs algorithm(s) that is/are correct but somewhat inefficient. | Student chooses/ designs efficient algorithm(s). | Student researches tradeoffs of different algorithms making sure the results is accurate and relevant. |  |
| **Design of logic (Flowchart / Pseudocode)** | No evidence of any logic by student. | Student’s work shows incomplete understanding on logic design. | Program has significant logic errors. | Program has slight logic errors that do not significantly affect the results. | Program is logically well designed. |  |
| **Designs Appropriate User Interface** | Implements very poor I/O functionality. | Only implements basic I/O functionality. | Some concepts of ‘user-friendly’ I/O used (e.g. prompts on input & labels on output). | Uses well-designed ‘user-friendly’ I/O interface appropriate for task and client. | User-friendly’ I/O interface (with GUI components - optional). |  |
| **Tests Program for Correctness** | No evidence of any testing by student. | Evidence of only one case tested. | Evidence of a few cases tested. | Evidence of “typical cases tested, but only assuming valid inputs. | ‘Robust design’ with extensive testing. |  |
| Documentation | Documentation merely contains the cover page and printout of the program code; no referencing. | Poor layout / flow; Missing some essential components within the documentation; Basic documentation standards not adhered to; No referencing. | Average layout / flow; No missing components of the documentation; Did some referencing but did not adhere to Harvard Name Referencing; Sample outputs available without any explanation. | Good layout / flow; No missing components of the documentation; Good documentation standards; Adhered to Harvard Name Referencing standards but with minor errors / omissions; Sample outputs available with some explanation. | Excellent layout / flow; No missing components of the documentation; Excellent documentation standards; Adhered to Harvard Name Referencing standards with no obvious errors / omissions; Sample outputs available with clear explanation. |  |

Remarks:

**ICT2102 Assignment 1 Rubrics**

Individual Component

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Group Marks: \_\_\_\_\_\_/ 50 Individual Marks: \_\_\_\_\_\_/ 30 Overall Marks: \_\_\_\_\_\_/ 80**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Criteria | Performance | | | | | Marks Given |
| Poor (0 – 2 marks) | Fair (3 – 4 marks) | Good (5 – 6 marks | Very Good (7 – 8 marks) | Excellent (9 – 10 marks) |
| Presentation | Did not turn up for presentation; Did not know how to execute the system. | Barely able to explain the codes / work done; Had difficulty in executing the system. | Able to explain some codes / work done; Able to execute the system; Only able to explain own component of the system / solution. | Provided good explanation of the codes and work done; Able to execute the system; Able to explain how own component of the system works with at least one other component of the system / solution. | Provided excellent explanation of the codes / work done; Able to execute the system; Able to explain how own component of the system works with all other components of the system / solution; Able to show additional concepts / new ideas used in the solution. |  |
| Question and Answer | Unable to provide answers to any of the questions asked. | Able to answer a few questions posed; Mostly inaccurate / illogical answers / explanation provided. | Able to answer some questions posed; Some accurate / logical answers / explanation provided; Reasonable answers given but with some hesitation | Able to answer most questions posed; Mostly accurate / logical answers / explanation provided; Reasonable answers given. | Able to answer all questions posed;  Accurate / logical answers / explanation provided; Reasonable answers given with sound arguments and clear discussion. |  |
| Contribution | Did not contribute at all to the project. | Minimal contribution to the project. | Average contribution to the project | Good contribution to the project. | Contributed the most to the project. |  |

Remark:

**ICT2102 Assignment 1 Rubrics**

Individual Component

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Group Marks: \_\_\_\_\_\_/ 50 Individual Marks: \_\_\_\_\_\_/ 30 Overall Marks: \_\_\_\_\_\_/ 80**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Criteria | Performance | | | | | Marks Given |
| Poor (0 – 2 marks) | Fair (3 – 4 marks) | Good (5 – 6 marks | Very Good (7 – 8 marks) | Excellent (9 – 10 marks) |
| Presentation | Did not turn up for presentation; Did not know how to execute the system. | Barely able to explain the codes / work done; Had difficulty in executing the system. | Able to explain some codes / work done; Able to execute the system; Only able to explain own component of the system / solution. | Provided good explanation of the codes and work done; Able to execute the system; Able to explain how own component of the system works with at least one other component of the system / solution. | Provided excellent explanation of the codes / work done; Able to execute the system; Able to explain how own component of the system works with all other components of the system / solution; Able to show additional concepts / new ideas used in the solution. |  |
| Question and Answer | Unable to provide answers to any of the questions asked. | Able to answer a few questions posed; Mostly inaccurate / illogical answers / explanation provided. | Able to answer some questions posed; Some accurate / logical answers / explanation provided; Reasonable answers given but with some hesitation | Able to answer most questions posed; Mostly accurate / logical answers / explanation provided; Reasonable answers given. | Able to answer all questions posed;  Accurate / logical answers / explanation provided; Reasonable answers given with sound arguments and clear discussion. |  |
| Contribution | Did not contribute at all to the project. | Minimal contribution to the project. | Average contribution to the project | Good contribution to the project. | Contributed the most to the project. |  |

Remark:

**ICT2102 Assignment 1 Rubrics**

Individual Component

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Group Marks: \_\_\_\_\_\_/ 50 Individual Marks: \_\_\_\_\_\_/ 30 Overall Marks: \_\_\_\_\_\_/ 80**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Criteria | Performance | | | | | Marks Given |
| Poor (0 – 2 marks) | Fair (3 – 4 marks) | Good (5 – 6 marks | Very Good (7 – 8 marks) | Excellent (9 – 10 marks) |
| Presentation | Did not turn up for presentation; Did not know how to execute the system. | Barely able to explain the codes / work done; Had difficulty in executing the system. | Able to explain some codes / work done; Able to execute the system; Only able to explain own component of the system / solution. | Provided good explanation of the codes and work done; Able to execute the system; Able to explain how own component of the system works with at least one other component of the system / solution. | Provided excellent explanation of the codes / work done; Able to execute the system; Able to explain how own component of the system works with all other components of the system / solution; Able to show additional concepts / new ideas used in the solution. |  |
| Question and Answer | Unable to provide answers to any of the questions asked. | Able to answer a few questions posed; Mostly inaccurate / illogical answers / explanation provided. | Able to answer some questions posed; Some accurate / logical answers / explanation provided; Reasonable answers given but with some hesitation | Able to answer most questions posed; Mostly accurate / logical answers / explanation provided; Reasonable answers given. | Able to answer all questions posed;  Accurate / logical answers / explanation provided; Reasonable answers given with sound arguments and clear discussion. |  |
| Contribution | Did not contribute at all to the project. | Minimal contribution to the project. | Average contribution to the project | Good contribution to the project. | Contributed the most to the project. |  |

Remark: