CCP REPORT

GROUP MEMBERS:

- MISHAL KHALID CT-25156
- DIYA RANI CT-25165

COURSE INSTRUCTOR: SIR ABDULLAH

COURSE TITLE: PROGRAMMING FUNDAMENTALS

COURSE CODE: CT-175

DISCIPLINE: BCIT

INSTITUTION: NED UNIVERSITY OF ENGINEERING &TECHNOLOGY

Index

PROJECT TITLE: THE HANGMAN GAME

ABSTRACT

This project presents the design and development of a Hangman Game in C. The main goal of this program is to apply logic building and string manipulation through an interactive guessing game. The system architecture emphasizes code organization through the use of user-defined functions, efficient string manipulation, and dynamic control structures that manage the flow of gameplay. The game selects a random food item from a predefined list and challenges the player to guess the word letter by letter within a limited number of chances. The implementation makes use of arrays, loops, conditional statements, and functions to manage game flow and display progress to the player. Text-based visual feedback enhances user engagement, while modular design and standard libraries demonstrate fundamental programming skills. Overall, the project showcases how logic-driven design and creative programming can turn a traditional word-guessing game into a challenging, engaging and intelligent digital experience.

INTRODUCTION

The Hangman Game project is a console-based interactive word-guessing system developed in the C programming language. It has been designed to integrate multiple programming concepts into a single, cohesive application that demonstrates algorithmic design, logical processing, and structured code organization. The project focuses on enhancing both computational thinking and practical programming by simulating a fully interactive game environment where the player must identify a secret word within a limited number of attempts.

The system architecture of the Hangman Game is based on a modular approach, dividing the program into independent sections that handle initialization, input, validation, and output. The main function controls the overall game flow—initializing variables, generating random words, and managing user interaction—while the user defined Hangman( ) function visualizes the game's progress according to the player's remaining chances. The use of standard C libraries such as <stdlib.h>, <string.h>, and <time.h> supports randomization, string manipulation, and timing operations that make the gameplay dynamic and unpredictable.

A key element of the program is the implementation of logic-based decision-making through conditional statements and looping constructs. These enable continuous gameplay until either the word is correctly guessed or all chances are exhausted. The game regularly updates the hidden word display, validating each user input while maintaining accurate tracking of progress and remaining attempts. Through efficient use of arrays and string comparison, the system ensures correct matching and error handling without performance loss.

This project serves as a strong representation of applied programming design—demonstrating how fundamental computational concepts such as control structures, randomization, and string operations can be combined to build a functional and engaging application. The Hangman Game reflects precision in code structure, effective logic handling, and well-coordinated interaction between functions, resulting in a program that is both technically sound and user-centered in design.

OBJECTIVES

- Implement a Hangman word-guessing game in C.
- Develop understanding of loops, conditionals, and arrays.
- Practice modular programming using functions.
- Enhance problem-solving skills and logical thinking.

TOOLS AND TECHNOLOGIES

- Operating System: Windows
- Language: C
- IDE: Dev-C++
- Libraries Used: <stdio.h>, <stdlib.h>, <string.h>, <time.h>

METHODOLOGY

1. Create a list of food-related words.
2. Randomly select a word using rand() seeded with time().
3. Represent the unguessed word with underscores '_'.
4. Take single-letter guesses from the user in a loop until all letters are guessed or chances run out.
5. Display Hangman figure based on remaining chances.
6. Track correct and incorrect guesses and update the word regularly.

EXPECTED OUTCOMES

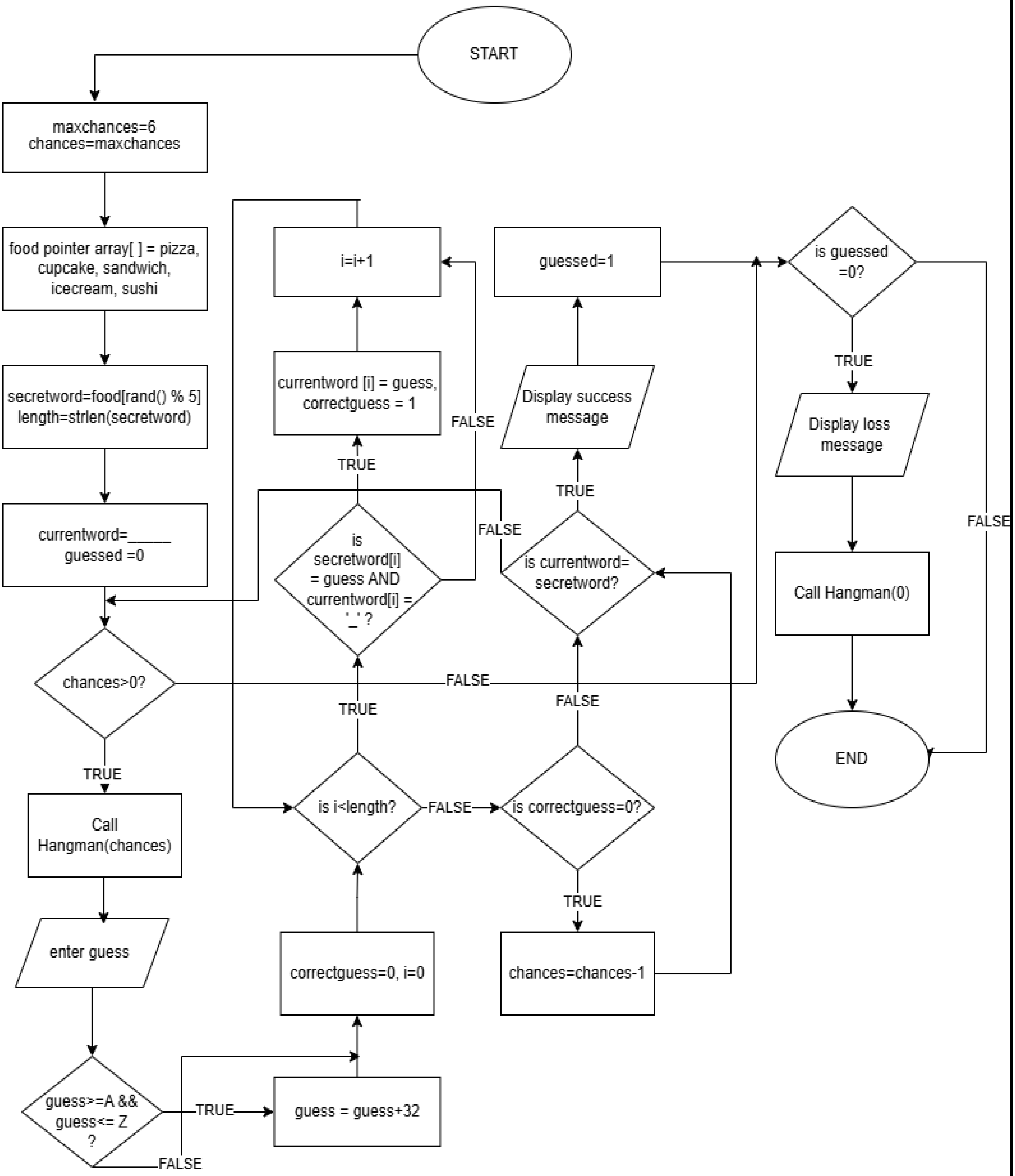
- Players can successfully guess the secret word within limited attempts.
- The game will display ASCII Hangman stages for each incorrect guess.
- Learning modular programming, loops, and string manipulation effectively.

PSEUDOCODE

START

```
BEGIN MAIN

    DISPLAY welcome message to player
    SET maxchances = 6
    DISPLAY game instructions and total number of chances
    DECLARE pointer array=["pizza","cupcake","sandwich","icecream”,"sushi"]

    CALL function srand(time(NULL)) to seed random number generator
    SET secretword = food[randomnumber % 5]
    SET chances = maxchances
    SET length = length of secretword
    DECLARE currentword [length + 1]

    FOR i= 0 TO length-1
        SET currentword[i] = '_'
    END FOR

    currentword[length] = '\0'
    SET flag guessed = FALSE

    WHILE chances > 0
        DISPLAY progress of currentword
        CALL Hangman(chances) to draw the current hangman figure
        PROMPT player to enter a guess letter
        READ guess
        IF guess is uppercase
            CONVERT it to lowercase to make game case-insensitive
       ENDIF
        SET correctguess = FALSE

        FOR i = 0 TO length-1
            IF secretword[i] == guess AND current_word[i] == '_'
                Add guessed letter in currentword
                SET correctguess = TRUE
            END IF
        END FOR

       IF correctguess == FALSE
            Decrease chance by 1
            DISPLAY wrong guess message and remaining chances
        END IF

        IF currentword matches secretword
            DISPLAY success message and the secretword
            SET guessed = TRUE
            BREAK WHILE
        END IF

    END WHILE

    IF guessed == FALSE
        DISPLAY message that player has lost and reveal secret word
        CALL function Hangman(0) to create the complete figure
    END IF
END MAIN


PROCEDURE Hangman(chancesleft)

    SWITCH chancesleft

        CASE 6:
            DISPLAY base structure only
        CASE 5:
            DISPLAY base structure with head
        CASE 4:
            DISPLAY base structure with head and torso
        CASE 3:
            DISPLAY base structure with head, torso and left arm
        CASE 2:
            DISPLAY base structure with head, torso and both arms
        CASE 1:
            DISPLAY base structure with head, torso, both arms and left leg
        CASE 0:
            DISPLAY full hangman (head, torso, both arms, and both legs)

    END SWITCH

END PROCEDURE

END
```

FLOWCHART


START
maxchances=6
chances=maxchances
food pointer array[ ] = pizza, cupcake, sandwich, icecream, sushi
secretword=food[rand() % 5]
length=strlen(secretword)
currentword=_____
guessed =0
chances>0?
TRUE
Call Hangman(chances)
enter guess
guess>=A && guess<= Z ?
TRUE
FALSE
guess = guess+32
correctguess=0, i=0
is i<length?
TRUE
is secretword[i] = guess AND currentword[i] = '_' ?
TRUE
currentword [i] = guess, correctguess = 1
i=i+1
FALSE
FALSE
is correctguess=0?
TRUE
chances=chances-1
FALSE
is currentword= secretword?
TRUE
Display success message
FALSE
guessed=1
FALSE
is guessed =0?
TRUE
Display loss message
Call Hangman(0)
FALSE
END

SOURCE CODE

```
#include <stdio.h>   // printf(), scanf()
#include <stdlib.h>  // rand(),srand()
#include <time.h>    // time()
#include <string.h>  // strlen(),strcpy(),strcmp()

void Hangman(int chances);  // function declaration
int main( )  // execution starts here
{
    printf("\n\t\t\t\t\tWelcome to the Hangman Game!\n\n");
    const int maxchances = 6;
    printf("Guess the secret food word!”);
    printf("You only have %d chances!\n”, maxchances);

    char *food[]={"pizza", "cupcake", "sandwich", "icecream", 
"sushi"};
    // food is a pointer array
    srand(time(NULL)); // seed random number generator
    char secretword[30];
    strcpy(secretword, food[rand() % 5]);//copy word to secretword
    int chances = maxchances;
    int length = strlen(secretword);
    char currentword[30];
    int i;

    for (i = 0; i < length; i++)
    {
        currentword[i] = '_'; // fill currentword with underscores
    }
    currentword[length] = '\0'; // add null character at the end
    char guess;
    int guessed = 0;

    while (chances > 0) // iterate until user has chances to guess
    {
        printf("\nWord: %s\n", currentword);
        Hangman(chances); // function call
        printf("\nEnter a letter: ");
        scanf(" %c", &guess); // saves input letter in guess
        if (guess >= 'A' && guess <= 'Z')
        {
            guess = guess + 32; // convert uppercase to lowercase
        }
        int correctguess = 0;
        for (i = 0; i < length; i++) // check if guess is correct
        {
            if (secretword[i] == guess && currentword[i] == '_')
            {
                currentword[i] = guess;
                correctguess = 1; // flag indicates guess was 
correct
            }
        }
        if (correctguess == 0) // if guess is wrong
        {
            chances--; // decrease chances
            printf("Wrong guess! Chances left: %d\n", chances);
        }
        else
        {
            printf("Correct guess!\n");
        }
        if(strcmp(currentword, secretword)==0)//check if guessed
        {
            printf("\nYou guessed it! The word is: %s\n", 
secretword);
            guessed = 1; // indicates word guessed
            break; // exit loop
        }
    }

    if (guessed == 0) // if chances = 0 and word not guessed
    {
        printf("\nGAME OVER!The word was: %s\n",secretword);
        Hangman(0); // complete hangman
    }
    return 0;
}


void Hangman(int chancesleft)
{
    switch (chancesleft)
    {
        case 6:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |      \n");
            printf(" |      \n");
            printf(" |      \n");
            printf(" |      \n");
            break;
        case 5:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |     O\n");
            printf(" |      \n");
            printf(" |      \n");
            printf(" |      \n");
            break;

        case 4:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |     O\n");
            printf(" |     |\n");
            printf(" |      \n");
            printf(" |      \n");
            break;
        case 3:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |     O\n");
            printf(" |    /|\n");
            printf(" |      \n");
            printf(" |      \n");
            break;

        case 2:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |     O\n");
            printf(" |    /|\\\n");
            printf(" |      \n");
            printf(" |      \n");
            break;
        case 1:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |     O\n");
            printf(" |    /|\\\n");
            printf(" |    / \n");
            printf(" |      \n");
            break;
        case 0:
            printf(" _______\n");
            printf(" |     |\n");
            printf(" |     |\n");
            printf(" |     O\n");
            printf(" |    /|\\\n");
            printf(" |    / \\\n");
            printf(" |        \n");
            break;
    }
}
```

TECHNICAL TERMS AND CONCEPTS:

This section explains the key technical terms and programming concepts used in the Hangman game implementation. These elements form the core of the code and illustrate essential aspects of C programming.

- **ARRAYS AND POINTERS**: Arrays are used to store the list of possible secret words. char *food[ ] = {"pizza", "cupcake", "sandwich", "icecream", "sushi"};) is an array of string pointers, where each element points to a constant string literal stored in memory. Pointers enable efficient memory access and manipulation, allowing random selection via indexing.

- **STRINGS AND STRING OPERATIONS**: Strings in C are null-terminated character arrays. Functions from <string.h> like strcpy() (to copy the secret word) and strcmp() (to compare the guessed word with the secret one) are crucial in this code. The currentword array tracks the partially revealed word, initialized with underscores ('_') to represent unknown letters. String length is determined using strlen() to iterate the loops appropriately.

- **RANDOM NUMBER GENERATION:** To ensure unpredictability, the code uses srand(time(NULL)) from <stdlib.h> and <time.h> to seed the random number generator with the current time. This prevents the same sequence of random numbers each time the program is run. The rand() % 5 expression selects a random index from the word array, introducing variability in gameplay.

- **LOOPS AND CONDITIONAL STATEMENTS:** A while loop controls the main game cycle, continuing as long as chances remain. Nested for loops handle word initialization and letter checking. Conditional structures like if and switch manage guess validation (e.g., converting uppercase to lowercase manually), correct/incorrect feedback, and hangman drawing based on remaining chances.

- **FUNCTIONS AND MODULARITY:** The code uses a modular approach with a main( ) function for overall execution and a separate user defined Hangman(int chances) function to draw the hangman figure. This separation promotes code reusability and organization. Function parameters allow dynamic behavior, while the switch statement in Hangman() handles different visual states.

- **INPUT/OUTPUT HANDLING:** Standard I/O functions from <stdio.h> such as printf( ) for displaying output and scanf( ) for reading input are used. The format scanf(" %c", &guess); includes a space to handle whitespace, ensuring reliable character input. Each case in the Hangman function uses multiple printf( ) calls to build the hangman figure progressively.

- **FLAGS AND VARIABLES:** Flags (e.g., int correctguess = 0; and int guessed = 0;) track states like whether a guess was accurate or if the word is fully revealed. Constants like const int maxchances = 6; enhance readability and ease modifications.

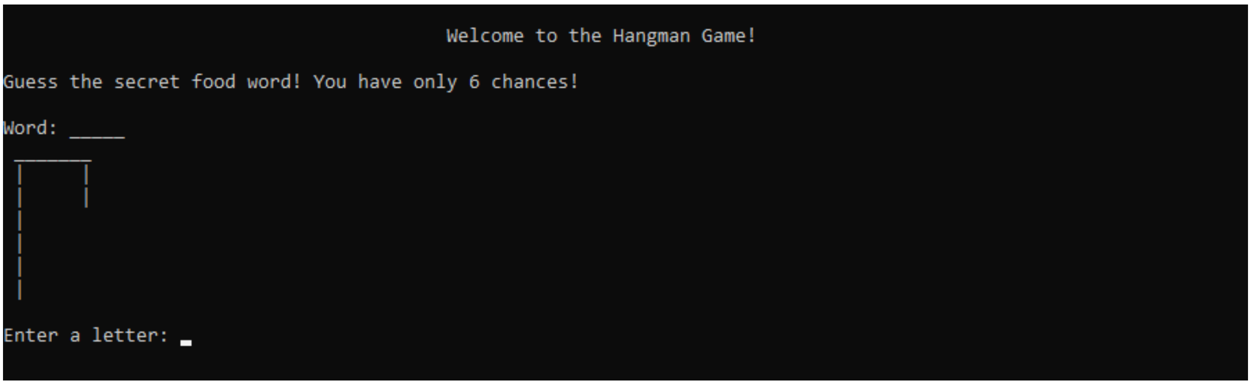
These concepts highlight C's procedural nature, emphasizing control flow, data structures, and library usage to build reliable and efficient systems.

Many beginners struggle to understand loops, arrays, and string handling in C. This project provides a fun and interactive way to apply these programming concepts in a practical game.

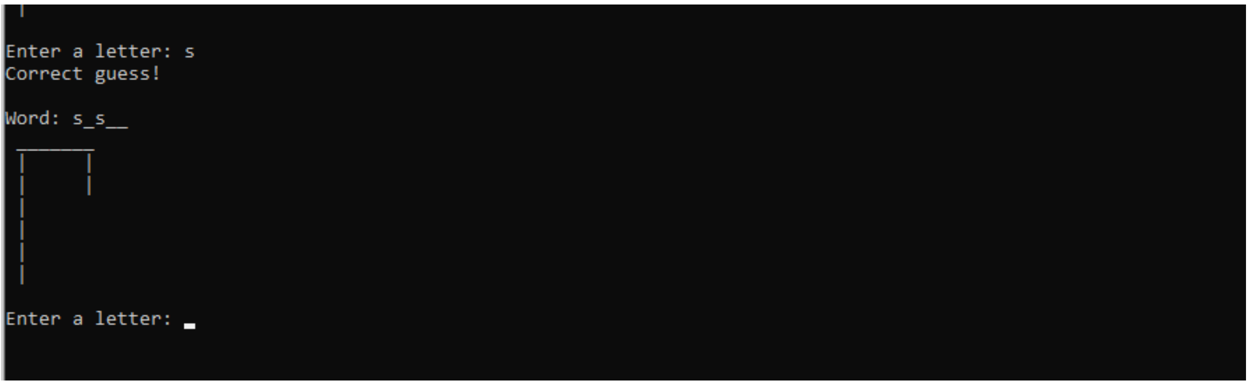
CONCLUSION

Through our Hangman Game project, we delivered a reliable, varied, and stable game that works perfectly every time. The use of clean structure, error-preventing techniques, and control shows real programming strength. The development of this project greatly improved our understanding of arrays, loops, conditional statements, and functions teaching us how to break down a problem into smaller steps and implement them efficiently and prepared us for tougher challenges ahead.

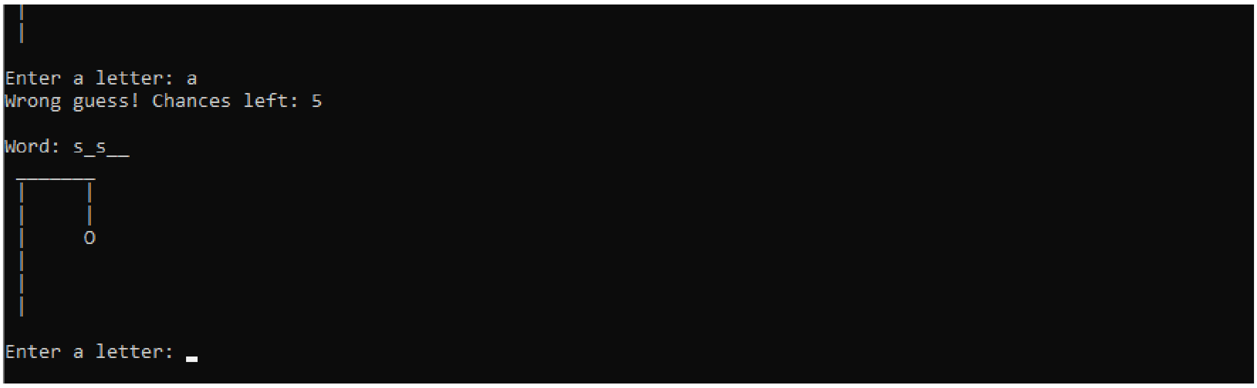
EXAMPLE RUN: PLAYER WINS!



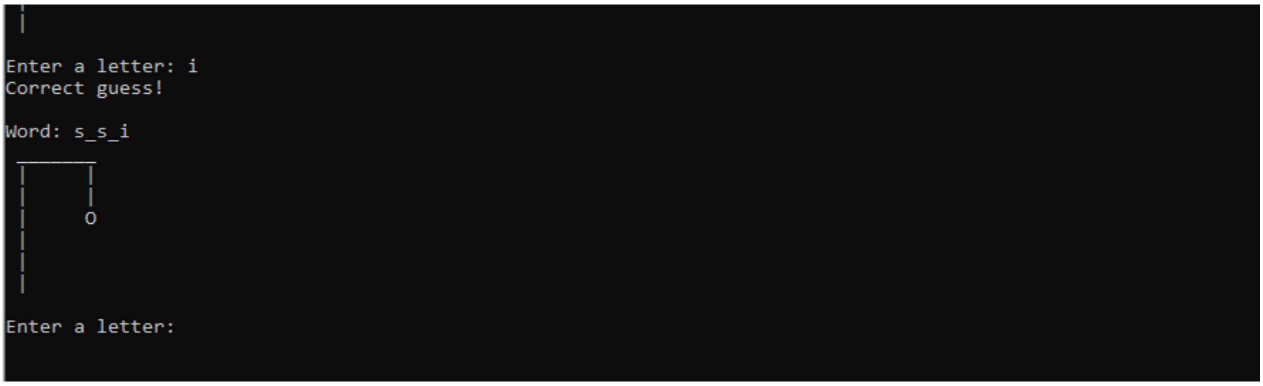
```
                                        Welcome to the Hangman Game!

Guess the secret food word! You have only 6 chances!

Word: _____
 _______
 |     |
 |     |
 |
 |
 |
 |

Enter a letter:
```



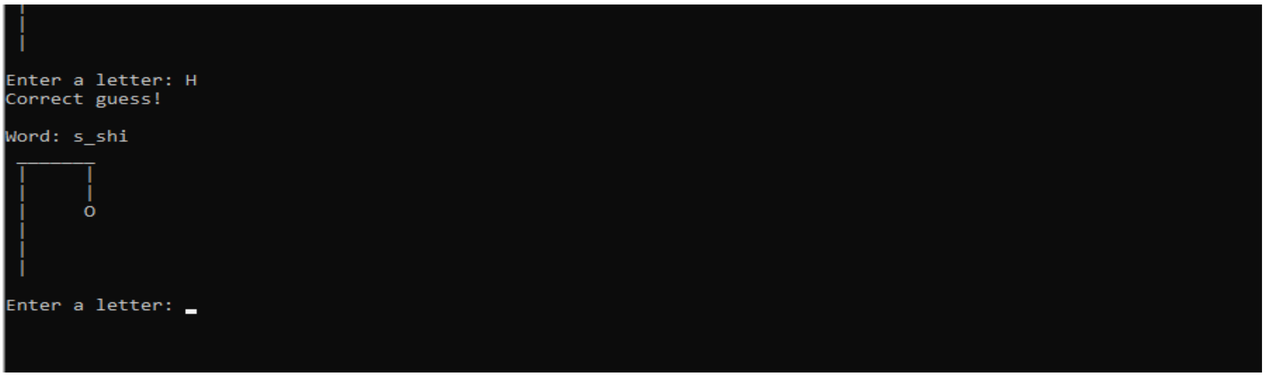
```
 |

Enter a letter: s
Correct guess!

Word: s_s__
 _______
 |     |
 |     |
 |
 |
 |
 |

Enter a letter:
```



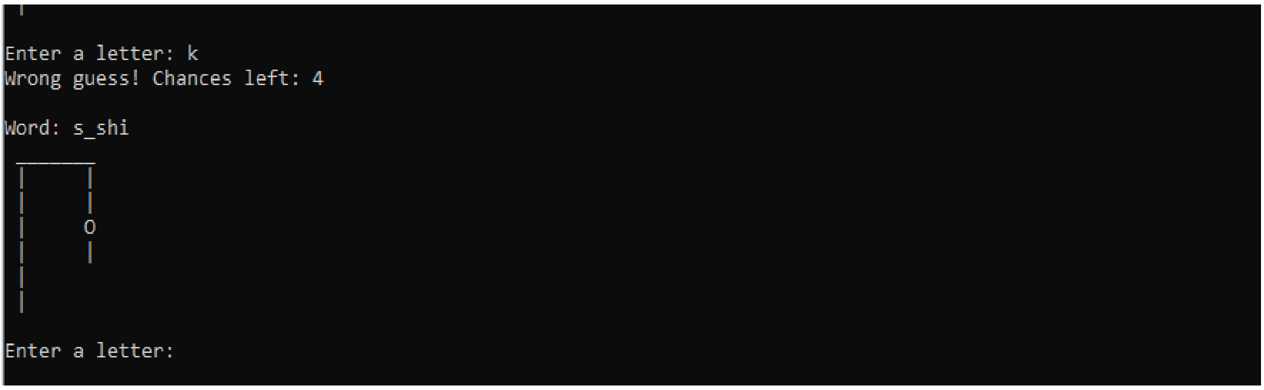
```
 |
 |

Enter a letter: a
Wrong guess! Chances left: 5

Word: s_s__
 _______
 |     |
 |     |
 |     O
 |
 |
 |

Enter a letter:
```



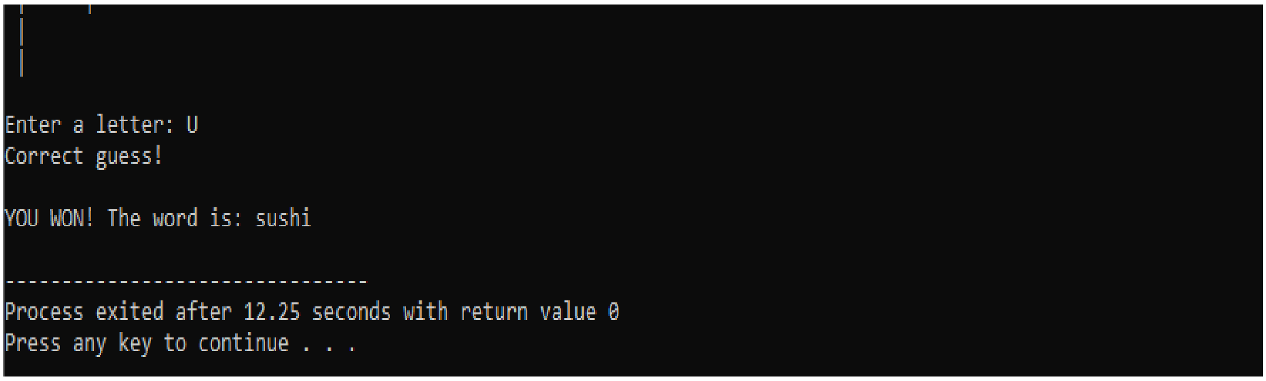
```
 |

Enter a letter: i
Correct guess!

Word: s_s_i
 _______
 |     |
 |     |
 |     O
 |
 |
 |

Enter a letter:
```



```
 |
 |

Enter a letter: H
Correct guess!

Word: s_shi
 _______
 |     |
 |     |
 |     O
 |
 |
 |

Enter a letter: _
```

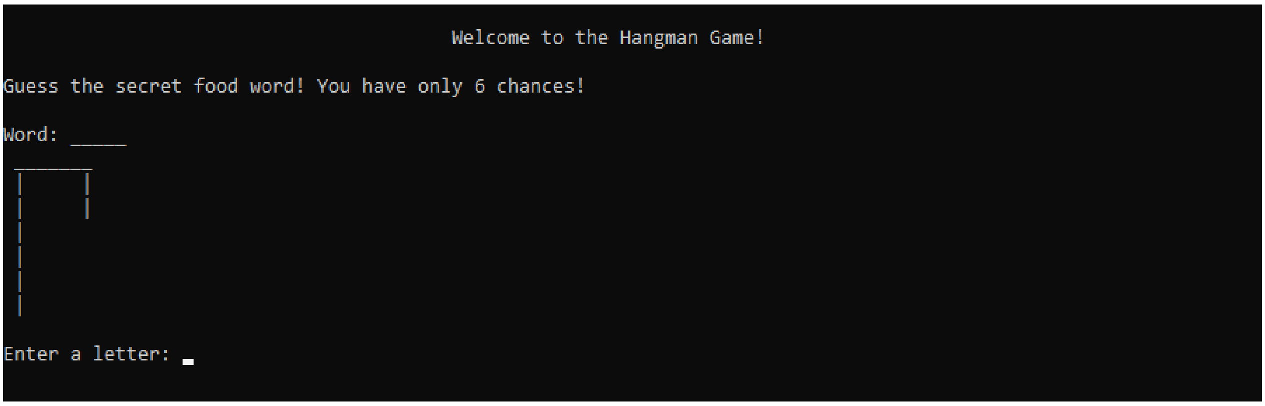


```
 |
Enter a letter: k
Wrong guess! Chances left: 4

Word: s_shi
 _______
 |     |
 |     |
 |     O
 |     |
 |
 |

Enter a letter:
```

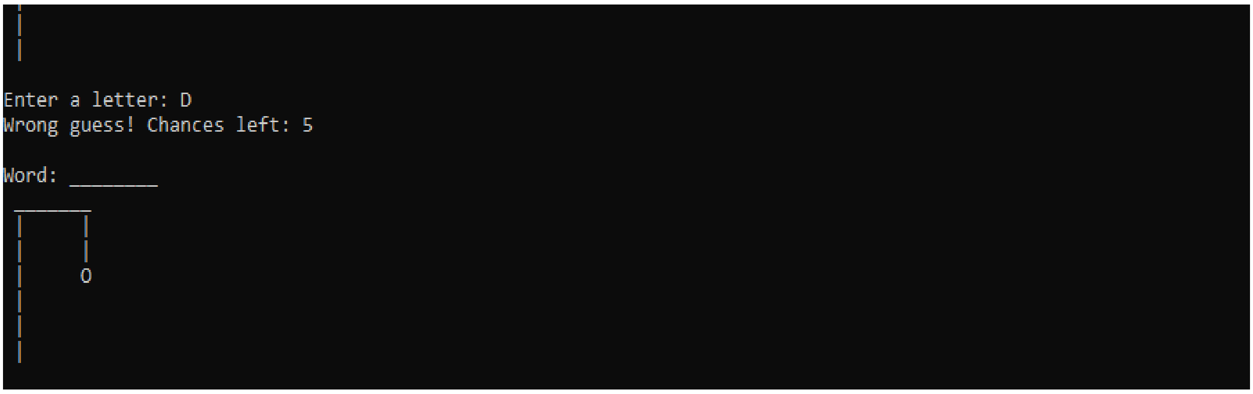


```
 |     |
 |
 |

Enter a letter: U
Correct guess!

YOU WON! The word is: sushi

--------------------------------
Process exited after 12.25 seconds with return value 0
Press any key to continue . . .
```

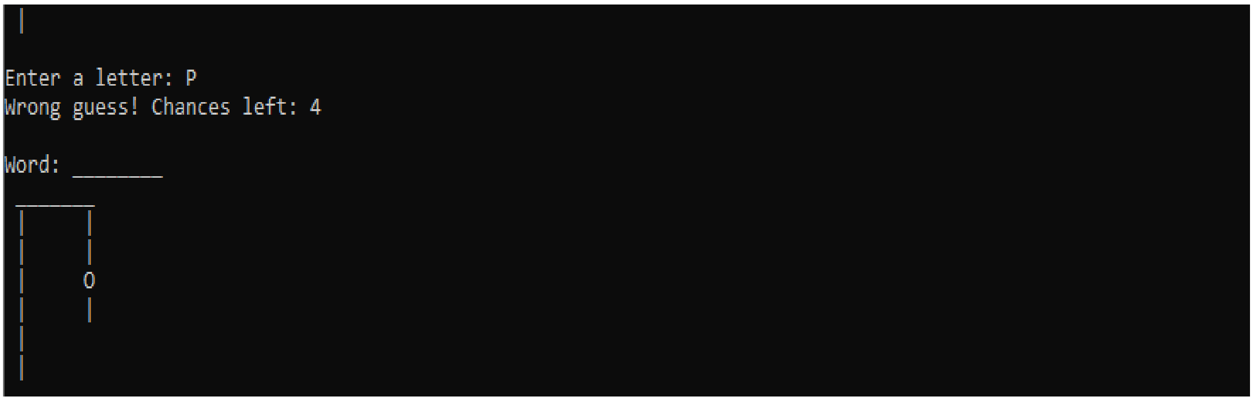
EXAMPLE RUN: PLAYER LOSES!



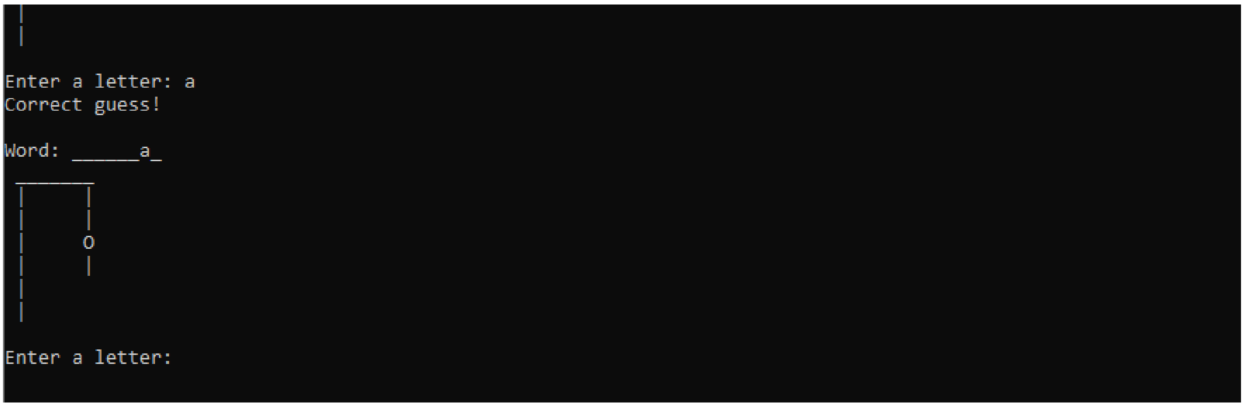
```
                                        Welcome to the Hangman Game!

Guess the secret food word! You have only 6 chances!

Word: _____
 _______
 |     |
 |     |
 |
 |
 |
 |

Enter a letter: _
```



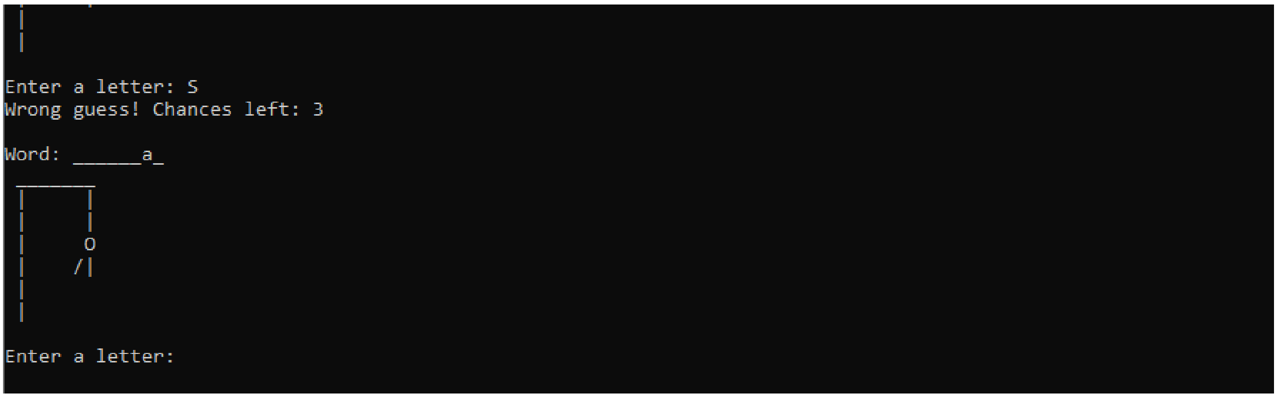
```
 |
 |

Enter a letter: D
Wrong guess! Chances left: 5

Word: ________

 _______
 |      |
 |      |
 |      O
 |
 |
 |
```



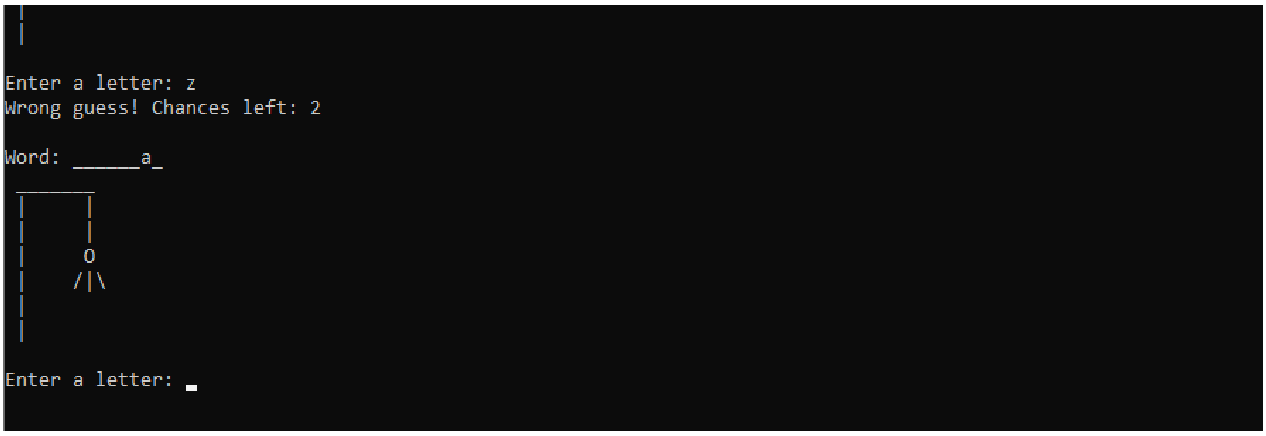
```
 |

Enter a letter: P
Wrong guess! Chances left: 4

Word: ________

 _______
 |      |
 |      |
 |      O
 |      |
 |
 |
```



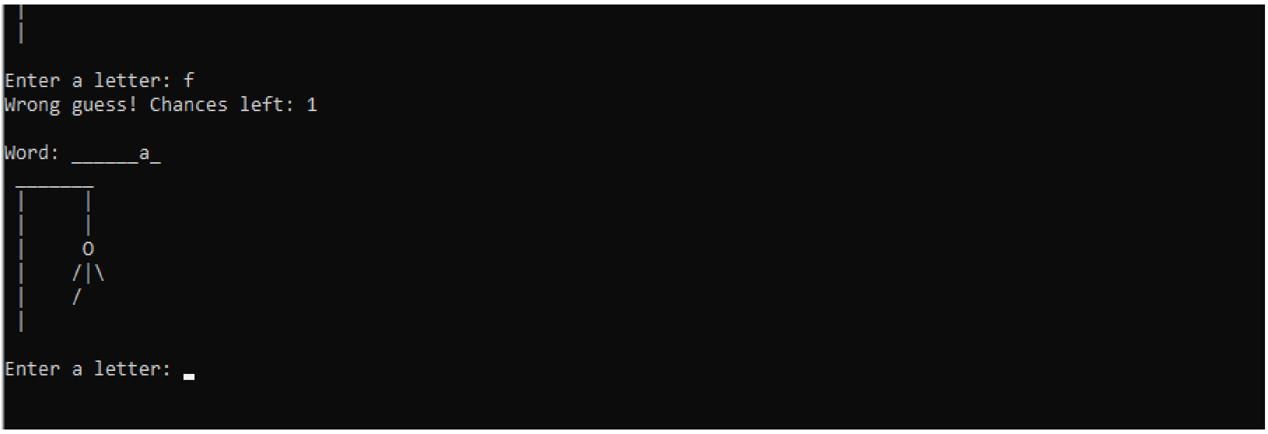
```
 |
 |

Enter a letter: a
Correct guess!

Word: ______a_

 _______
 |      |
 |      |
 |      O
 |      |
 |
 |

Enter a letter:
```



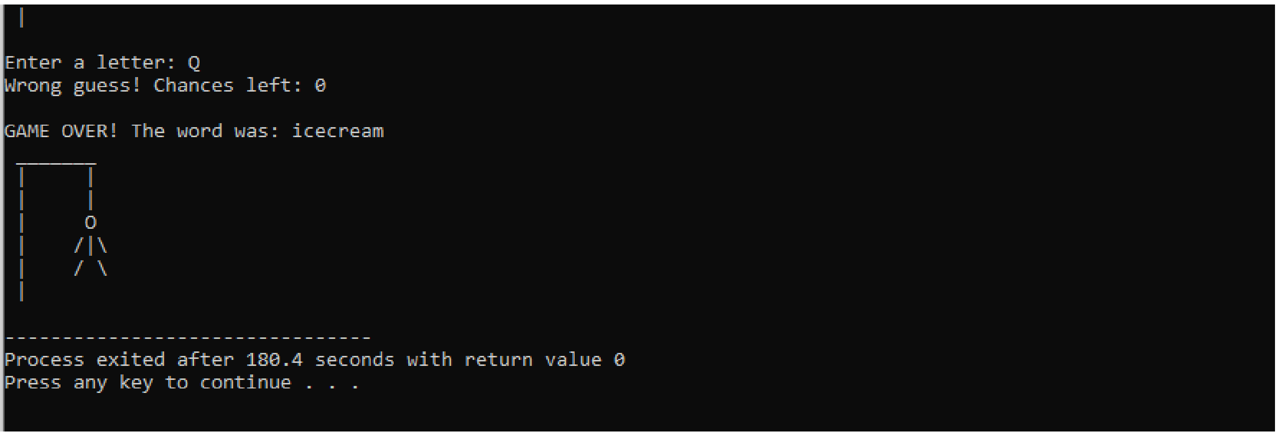
```
 |
 |

Enter a letter: S
Wrong guess! Chances left: 3

Word: ______a_

 _______
 |      |
 |      |
 |      O
 |     /|
 |
 |

Enter a letter:
```



```
 |
 |

Enter a letter: z
Wrong guess! Chances left: 2

Word: ______a_
 _______
 |     |
 |     |
 |     O
 |    /|\
 |
 |

Enter a letter: 
```



```
 |
 |

Enter a letter: f
Wrong guess! Chances left: 1

Word: ______a_
 _______
 |     |
 |     |
 |     O
 |    /|\
 |    /
 |

Enter a letter: 
```



```
 |

Enter a letter: Q
Wrong guess! Chances left: 0

GAME OVER! The word was: icecream
 _______
 |     |
 |     |
 |     O
 |    /|\
 |    / \
 |

--------------------------------
Process exited after 180.4 seconds with return value 0
Press any key to continue . . .
```