TECNIA INSTITUTE OF ADVANCED STUDIES
GRADE "A" INSTITUTE
Approved by AICTC, Ministry of HRD, Govt. of India, Affiliated to GGSIP University
Recognized Under Sec. 2(f) of UGC Act 1956
INSTITUTIONAL AREA MADHUBAN CHOWK,ROHINI,DELHI 110085
Tel:91-11-27555121-24, E-Mail : directortias@tecnia.in, Website: www.tiaspg.tecnia.in

EGAC Accredited 21001:2018 51001:2018

DEPARTMENT OF INFORMATION COMMUNICATION & TECHNOLOGY

MINI PROJECT OF BCA-106

Data Structure & Algorithm

Academic Session: 2024-25

Batch: 2024-28

Submittedto:
Mr. Sukant Vats
(Subject-In-Charge)

Submitted by:
Name: Preeti Raj
Enrolment No: 06617002024
Program: BCA
Semester: 2nd
Shift: Morning
Division: B

TECNIA INSTITUTE OF ADVANCED STUDIES
GRADE "A" INSTITUTE
Approved by AICTC, Ministry of HRD, Govt. of India, Affiliated to GGSIP University
Recognized Under Sec. 2(f) of UGC Act 1956
INSTITUTIONAL AREA MADHUBAN CHOWK,ROHINI,DELHI 110085
Tel:91-11-27555121-24, E-Mail : directortias@tecnia.in, Website: www.tiaspg.tecnia.in

EGAC Accredited 21001:2018 51001:2018

DEPARTMENT OF INFORMATION COMMUNICATION & TECHNOLOGY

Project Title:	Number Guessing Game
Subject:	Data Structure & Algorithm
Session:	2024-25
Submitted to:	Mr. Sukant Vats
Program & Sem:	BCA II-SEMESTER
Submitted by:	Preeti Raj
Software Used:	Visual Studio

ABSTRACT

.

This project presents a console-based Number Guessing Game developed in C programming language, designed to showcase the practical implementation of **Data Structures and Algorithms (DSA)** — particularly the **Queue** data structure implemented using a **singly linked list**.

The game involves the system generating a random number, and the user attempts to guess it. Each incorrect guess is stored in a queue to track the history of user attempts, thereby demonstrating the **First In First Out (FIFO)** behaviour of a queue. The game features multiple difficulty levels, hints to guide the player, and the ability to quit at any time. This project emphasizes logic building, modular design, and memory management using dynamic allocation.

INTRODUCTION

In the realm of programming, games are often used as a fun and interactive way to understand and apply core concepts. This project focuses on the use of the **Queue** data structure to maintain a record of user guesses in a guessing game scenario.

In this game:

- The user selects a difficulty level.
- The system generates a number within a specified range based on the level.
- The user attempts to guess the number.
- Each incorrect guess is stored in a queue (linked list).
- The user is provided hints and can quit the game by entering -1.

The project merges logic-building exercises with DSA implementation, offering a meaningful way to reinforce both.

Key Features :

1) Difficulty Levels: Easy (0–100), Medium (0–1000), Hard (0–5000)
2) Intelligent Hints: Informs whether the guess should be higher or lower
3) Guess Tracker: Stores all incorrect guesses using a queue
4) Quit Option: Type -1 anytime to exit the game
5) Replay Mode: Users can choose to replay the game after each round
6) Memory Management: Uses malloc() to dynamically allocate memory for guesses
7) Display Guesses: Lists all previous wrong attempts at the end of the game

PROJECT OVERVIEW

1. **Queue Implementation**

   A linked list is used to store guesses in the order they are made.

   - New nodes are added at the rear using the insert() function.
   - The display() function traverses and prints all wrong guesses.

2. **Random Number Generation**
   - rand() generates a random number.
   - srand(time(NULL)) ensures different numbers are generated every time the game runs.

3. **User Interaction**
   - Input is taken using scanf().
   - If the guess is incorrect, it is enqueued.
   - If the user enters -1, the game terminates gracefully.
   - The game can be replayed by choosing an option at the end.

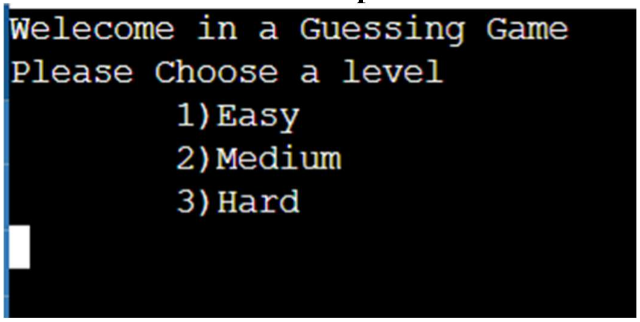
4. **Hints System**
   - After each wrong guess, the program suggests whether to guess higher or lower.

CODE

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

struct guess{
    int data;
    struct guess *next;
};

struct guess *front=NULL;
struct guess *rear=NULL;
int count=0;

void insert(int data){
    struct guess *new=(struct guess*)malloc(sizeof(struct guess));
    new->data=data;
    new->next=NULL;
    if(front==NULL){
        rear=new;
        front=new;
    }
    else{
        rear->next=new;
        rear=new;
    }
}

void display(){
    struct guess *line=front;

    if(line==NULL){
        printf("\n\tYou guessed in Single Try\n\tBig Congratulation (^o^)\n");
    }
    else{
        printf("\nAll your guessed number\n");
        while(line != NULL){
            printf("%d\t",line->data);
            line=line->next;
        }
    }
}

void find(int y){
    int data;
    do{
        printf("Guess the number : ");
        scanf("%d",&data);
        if(data == -1){
            printf("You have chosen to quit this game");
            exit(1);
        }
        else if(data == y){
            printf("Congratulation , YOU WON\n");
            display();
            printf("\nWrong guesses : %d",count);
        }
        else{
            printf("\tOH NO!,Try again\n");
            insert(data);
            ++count;
            if(data>y){
                printf("\tGuess lower value\n");
            }
            else{
                printf("\tGuess higher value\n");
            }
        }
    }while(data != y);
}

void cleanup(){
    struct guess *x;
    while(front != NULL){
        x=front;
        front=front->next;
    }
}

int main(){

    int y;
    int level;
    printf("Welecome in a Guessing Game\n");

    here:
    printf("Please Choose a level\n");
    printf("\t1)Easy\n\t2)Medium\n\t3)Hard\n");
    scanf("%d",&level);
    switch(level){

        case 1:
            srand(time(NULL));
            y=rand();
            while(y>101){
                y%=101;
            }

            printf("Range 0:100\n");
            find(y);
            break;

        case 2:
            srand(time(NULL));
            y=rand();
            while(y>1001){
                y%=1001;
            }

            printf("Range 0:1000\n");
            find(y);
            break;

        case 3:
            srand(time(NULL));
            y=rand();
            while(y>5001){
                y%=5001;
            }

            printf("Range 0:5000\n");
            find(y);
    }

    char again;
    printf("\nWant to play again (y/n) : ");
    scanf(" %c",&again);

    cleanup();
    if(again == 'y'){
        goto here;
    }
    return 0;
}
```

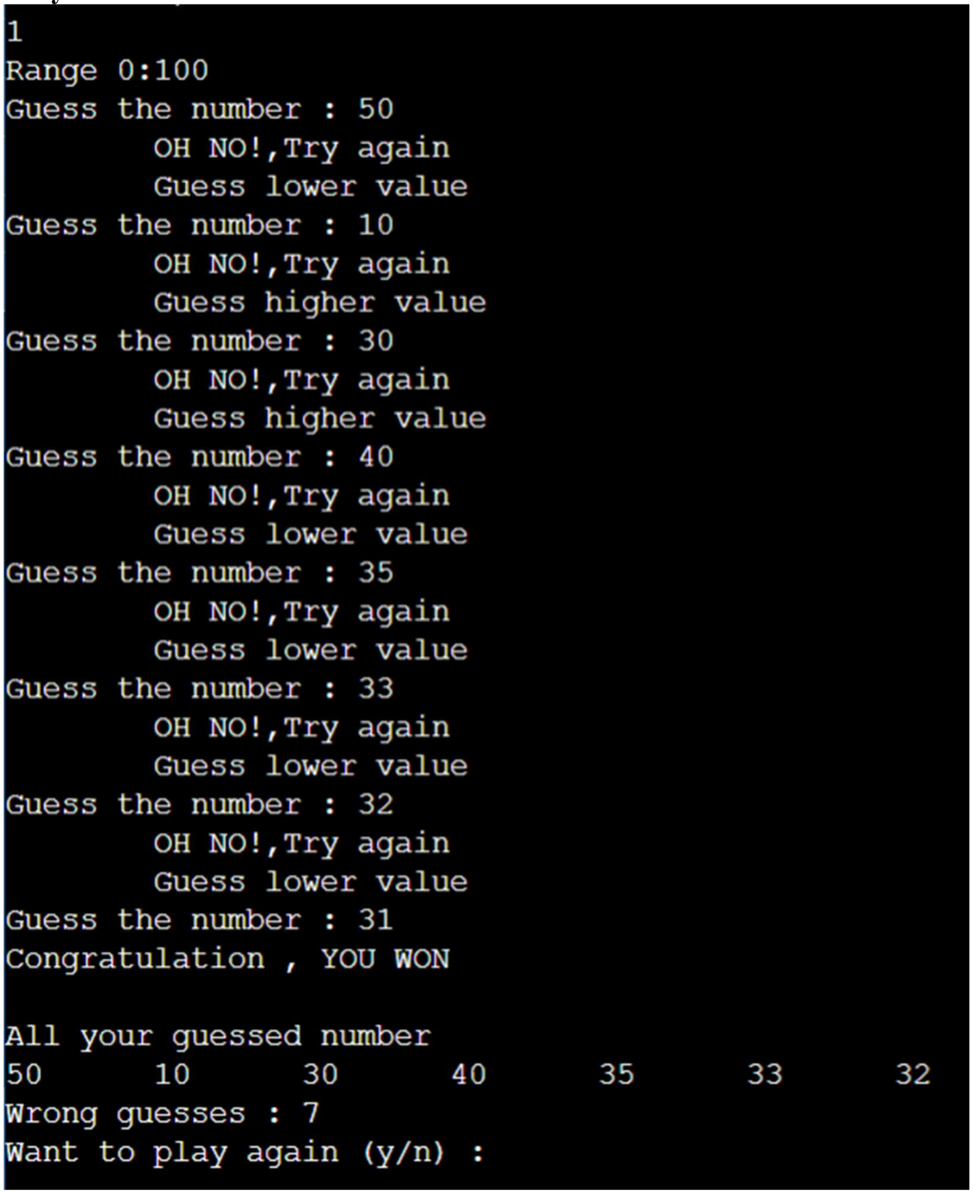
OUTPUT

Welcome screen and level option



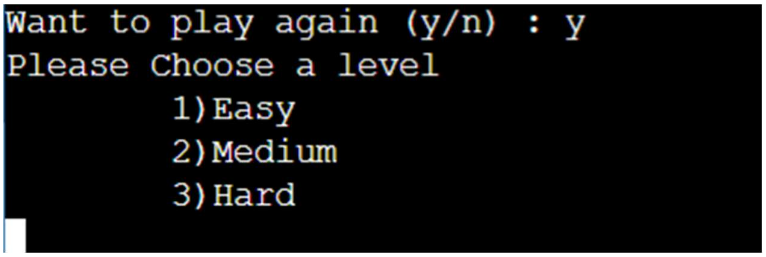
```
Welecome in a Guessing Game
Please Choose a level
         1)Easy
         2)Medium
         3)Hard
```

Easy Level



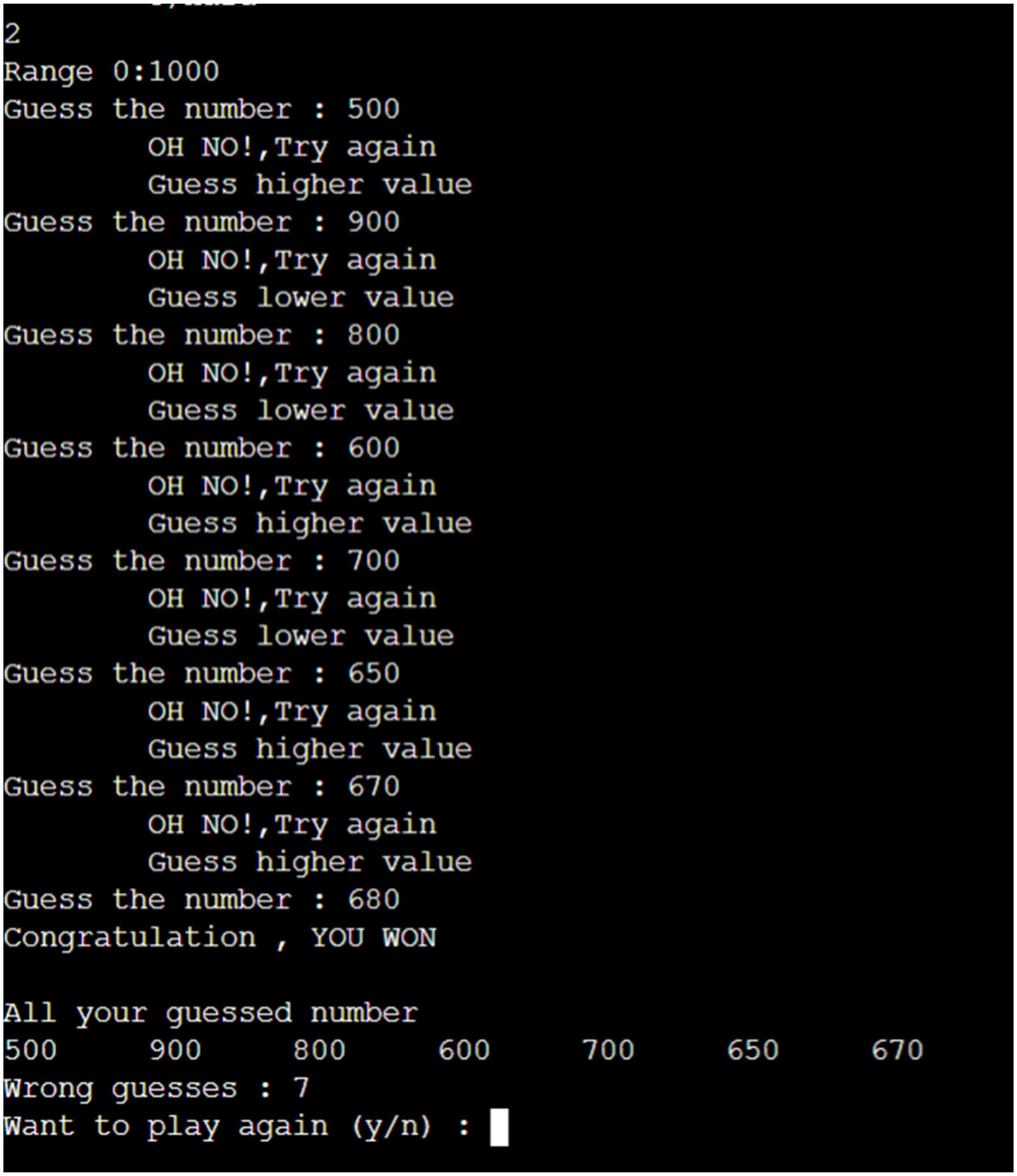
```
1
Range 0:100
Guess the number : 50
        OH NO!,Try again
        Guess lower value
Guess the number : 10
        OH NO!,Try again
        Guess higher value
Guess the number : 30
        OH NO!,Try again
        Guess higher value
Guess the number : 40
        OH NO!,Try again
        Guess lower value
Guess the number : 35
        OH NO!,Try again
        Guess lower value
Guess the number : 33
        OH NO!,Try again
        Guess lower value
Guess the number : 32
        OH NO!,Try again
        Guess lower value
Guess the number : 31
Congratulation , YOU WON

All your guessed number
50      10      30      40      35      33      32
Wrong guesses : 7
Want to play again (y/n) :
```

If user choose to play again



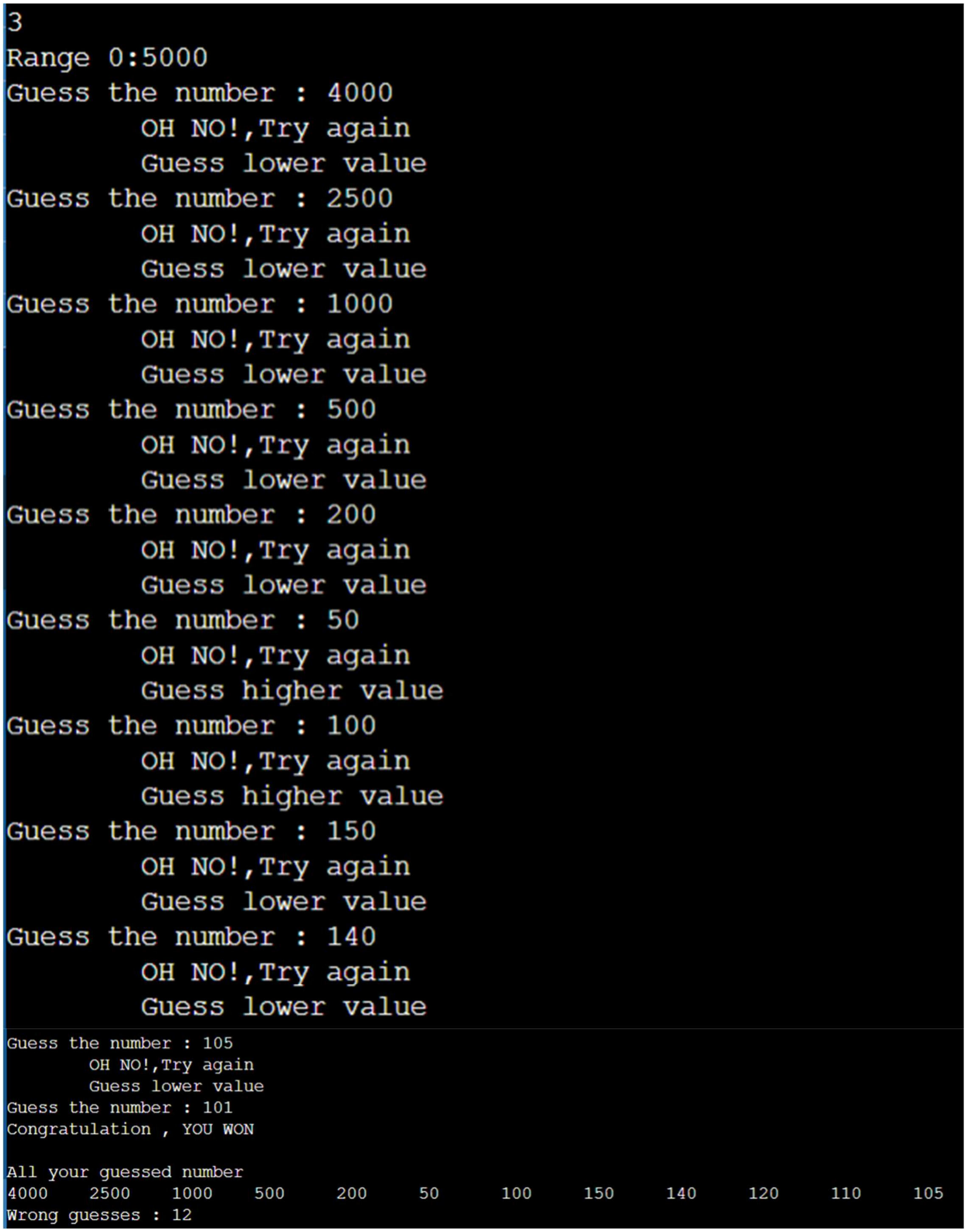
```
Want to play again (y/n) : y
Please Choose a level
        1)Easy
        2)Medium
        3)Hard
```

Medium level



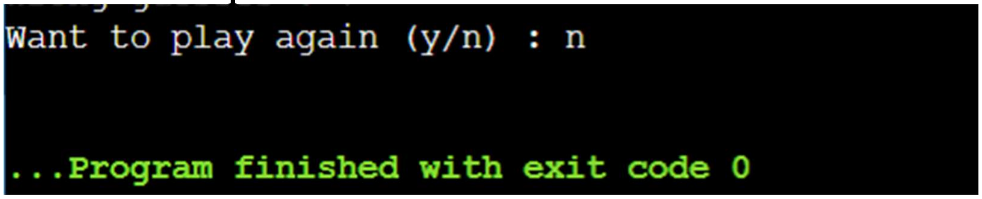
```
2
Range 0:1000
Guess the number : 500
        OH NO!,Try again
        Guess higher value
Guess the number : 900
        OH NO!,Try again
        Guess lower value
Guess the number : 800
        OH NO!,Try again
        Guess lower value
Guess the number : 600
        OH NO!,Try again
        Guess higher value
Guess the number : 700
        OH NO!,Try again
        Guess lower value
Guess the number : 650
        OH NO!,Try again
        Guess higher value
Guess the number : 670
        OH NO!,Try again
        Guess higher value
Guess the number : 680
Congratulation , YOU WON

All your guessed number
500     900     800     600     700     650     670
Wrong guesses : 7
Want to play again (y/n) :
```

Hard Level



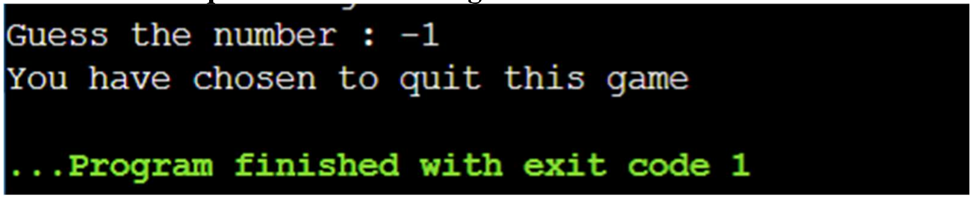
```
3
Range 0:5000
Guess the number : 4000
        OH NO!,Try again
        Guess lower value
Guess the number : 2500
        OH NO!,Try again
        Guess lower value
Guess the number : 1000
        OH NO!,Try again
        Guess lower value
Guess the number : 500
        OH NO!,Try again
        Guess lower value
Guess the number : 200
        OH NO!,Try again
        Guess lower value
Guess the number : 50
        OH NO!,Try again
        Guess higher value
Guess the number : 100
        OH NO!,Try again
        Guess higher value
Guess the number : 150
        OH NO!,Try again
        Guess lower value
Guess the number : 140
        OH NO!,Try again
        Guess lower value
Guess the number : 105
        OH NO!,Try again
        Guess lower value
Guess the number : 101
Congratulation , YOU WON

All your guessed number
4000    2500    1000    500     200     50      100     150     140     120     110     105
Wrong guesses : 12
```

If user choose to quit



```
Want to play again (y/n) : n


...Program finished with exit code 0
```

If user want to quit in middle of the game



```
Guess the number : -1
You have chosen to quit this game

...Program finished with exit code 1
```

LEARNING OUTCOMES

By completing this project, students will learn:

- Practical application of queue operations using a linked list
- Memory management with dynamic allocation
- Use of pointers for data structure manipulation
- Implementation of control flow using loops and conditions
- Effective use of random functions and user input handling
- Writing modular, reusable, and interactive code

CONCLUSION

This project not only introduces students to the fundamentals of C programming but also shows how **Data Structures**, such as Queues implemented with Linked Lists, can be applied in real-life situations. The Number Guessing Game serves as an engaging platform to practice DSA concepts, pointer manipulation, and logic development in a fun, user-friendly manner. It reinforces both theoretical and practical aspects of programming and data handling, making it a valuable beginner-to-intermediate level DSA project.

REFERENCES

- https://www.w3schools.com
- https://www.geeksforgeeks.org
- https://chat.openai.com