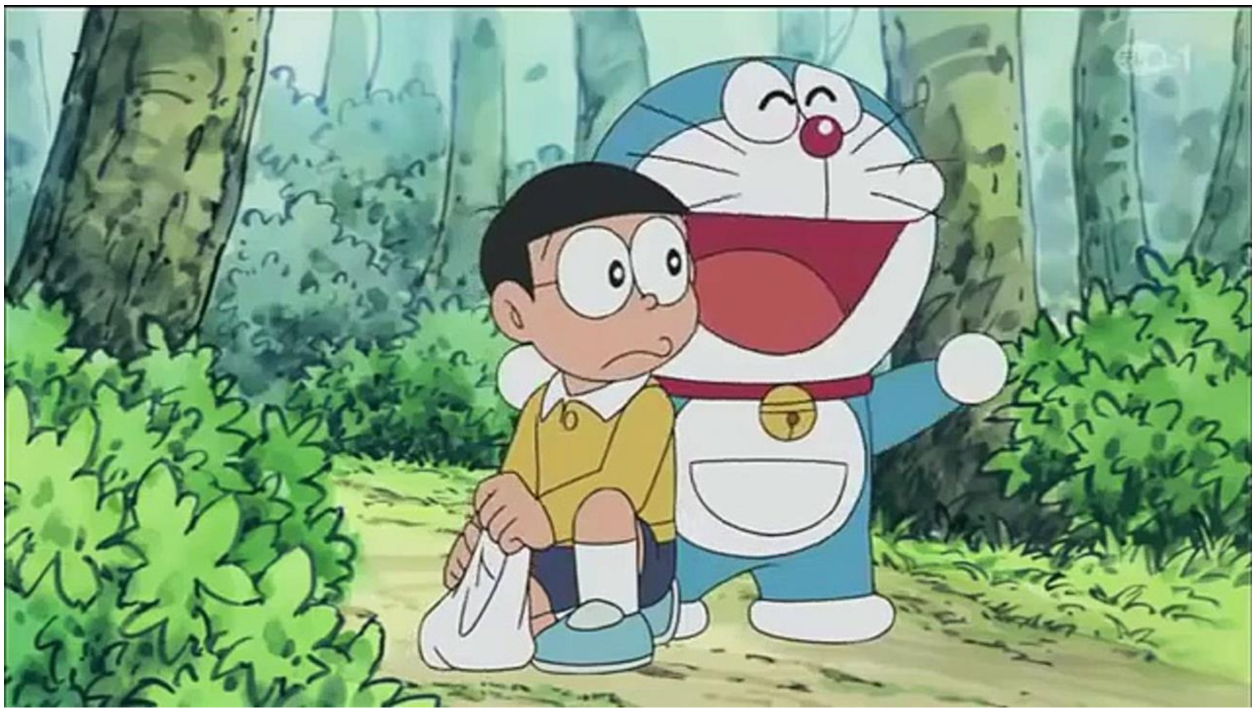
Nobita's Wild Adventure: A Jungle Escape



GROUP MEMBERS

BUSHRA ANSAR (AI-003)

TAMIA NAEEM (AI-004)

BUSHRA ATIQ (AI-025)

Table of Contents

INTRODUCTION

Project Statement

Nobita's Wild Adventure is an interactive text-based game that takes the player, Nobita, on a journey through the jungle. The game consists of three levels, each with unique challenges and hurdles that Nobita must overcome to reach his home. The challenges include guessing a secret number, solving anagrams, and playing rock-paper-scissors. Throughout the game, Nobita receives assistance from his trusty friend, Doraemon, who provides magical gadgets to help him overcome obstacles.

Project Description

Level 1: The Secret Number

Nobita starts by realizing he has lost his map in the jungle. To retrieve it, he must guess a secret number within 5 attempts. The player interacts by inputting their guesses, and Doraemon appears to assist after successfully guessing the number.

Level 2: Anagrams Challenge

Nobita encounters a river without a bridge and hears a mysterious sound. To cross, he needs to solve anagrams by inputting pairs of words. Doraemon arrives to help after successfully completing the anagrams challenge, providing a magical ship to cross the river.

Level 3: Rock-Paper-Scissors with a Dinosaur

A dinosaur challenges Nobita to a rock-paper-scissors game. The player inputs their choice, and the dinosaur's choice is randomly generated. Depending on the outcome, Nobita may win or lose. Doraemon intervenes after Nobita defeats the dinosaur, providing a magical torch to make the dinosaur small.

Final Challenge: Choosing the Right Door

After overcoming all hurdles, Nobita faces three doors. The player selects a door, and only one leads to Nobita's home. Correctly choosing the door concludes the game.

Group Members Contribution

- ***Tamia Naeem*** crafted the ASCII code for various elements, such as gadgets, Nobita, Doraemon, and hurdles. Additionally, she implemented the code for level 1 (number guessing game) and encapsulated them within functions.
- ***Bushra Atiq*** played a crucial role in the project by writing the code for level 2 (anagram), fine-tuning sleep functions, and all other functions and crafting dialogues between Nobita and Doraemon.
- ***Bushra Ansar*** significantly contributed to the project by providing the code for level 3 (Rock-Paper-Scissors) and developing the final challenge.

LIBRARIES

1. <stdio.h> (Standard Input/Output):

- Utilized for basic input/output operations like printing text (printf) and receiving user input (scanf).
- Enables communication between the program and the user via the console.

2. <stdlib.h> (Standard Library):

- Provides essential functions for random number generation (rand).
- Used to manage dynamic memory and introduce randomness into the game.

3. <time.h> (Time):

- Used for time-related operations, particularly for seeding the random number generator.
- Ensures that the random number sequence is different in each game.

4. <string.h> (String):

- Offers functions for string manipulation, such as copying strings (strcpy), comparing strings (strcmp), and finding string lengths (strlen).
- Used to handle and manipulate textual data in the game.

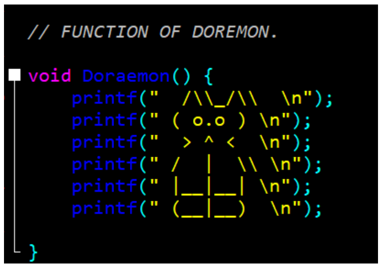
5. <unistd.h> (POSIX):

- Provides POSIX-compliant functions, including sleep, which introduces delays in the execution of the program.
- Used for creating pauses in the game, enhancing user experience.

FUNCTIONS

1. Doraemon():

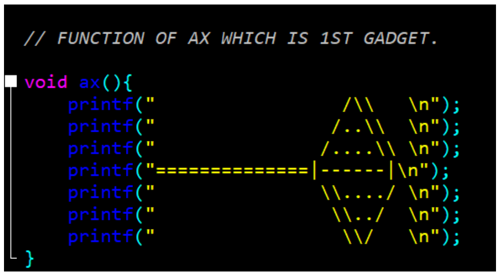
- Displays an ASCII art representation of Doraemon, contributing visual appeal to the game.
- Aesthetic function that enhances the overall presentation of the characters.



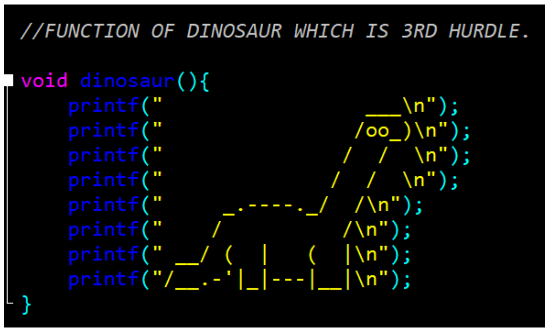
```c
// FUNCTION OF DOREMON.

void Doraemon() {
    printf("  /\\_/\\  \n");
    printf(" ( o.o ) \n");
    printf("  > ^ <  \n");
    printf(" /  |  \\ \n");
    printf(" |__|__| \n");
    printf(" (__|__)  \n");
}
```

2. ax(), water(), ship(), dinosaur(), torch(), tree():

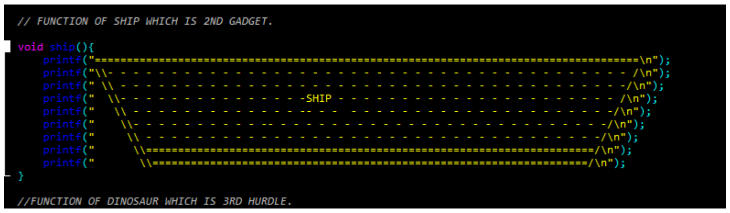
- Each function displays ASCII art representing various elements in the game, such as gadgets, hurdles, and characters.
- These functions contribute to the visual storytelling aspect of the game.



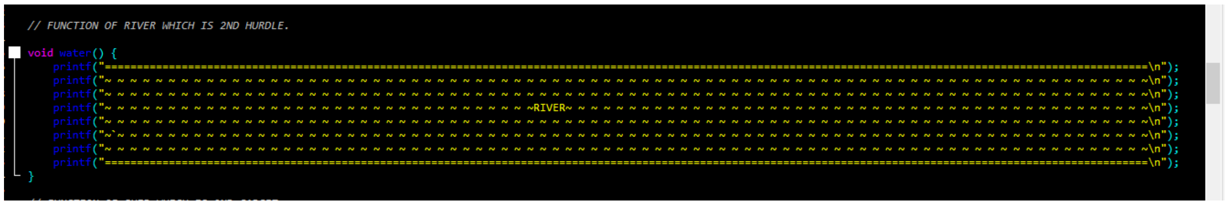
```c
// FUNCTION OF AX WHICH IS 1ST GADGET.

void ax(){
    printf("              /\\   \n");
    printf("             /..\\  \n");
    printf("            /....\\ \n");
    printf("==============|------|\n");
    printf("            \\..../ \n");
    printf("             \\../  \n");
    printf("              \\/   \n");
}
```

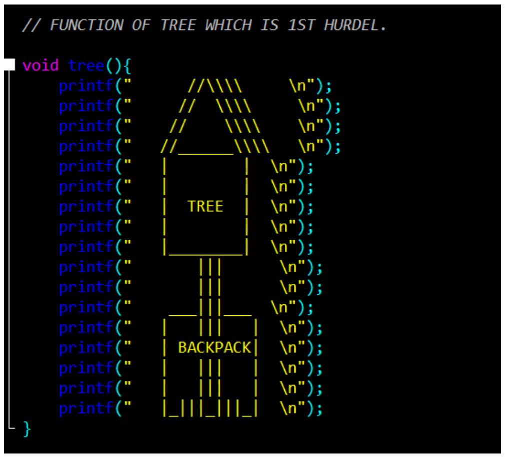


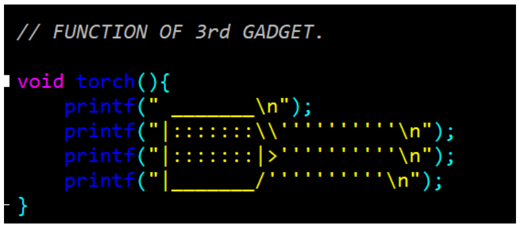
```c
//FUNCTION OF DINOSAUR WHICH IS 3RD HURDLE.

void dinosaur(){
    printf("                    ___\n");
    printf("                   /oo_)\n");
    printf("                  /  /  \n");
    printf("                 /  /  \n");
    printf("      _.----._/  /\n");
    printf("     /           /\n");
    printf(" __/ (  |   (  |\n");
    printf("/__.-'|_|---|__|\n");
}
```



```c
// FUNCTION OF SHIP WHICH IS 2ND GADGET.

void ship(){
    printf("=========================================================================\n");
    printf("\\- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - /\n");
    printf(" \\ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -/\n");
    printf("  \\- - - - - - - - - - - - - - - -SHIP - - - - - - - - - - - - - - - - /\n");
    printf("   \\ - - - - - - - - - - - - -- - -  - - - - - - - - - - - - - - - -/\n");
    printf("    \\- - - - - - - - - - -- - - - - - - - - - - - - - - - - - - - -/\n");
    printf("     \\ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -/\n");
    printf("      \\==============================================================/\n");
    printf("       \\============================================================/\n");
}

//FUNCTION OF DINOSAUR WHICH IS 3RD HURDLE.
```

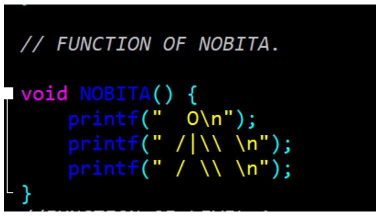

```c
// FUNCTION OF RIVER WHICH IS 2ND HURDLE.

void water() {
    printf("=====================================================================================================================================================================\n");
    printf("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");
    printf("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");
    printf("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~RIVER~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");
    printf("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");
    printf("~`~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");
    printf("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");
    printf("=====================================================================================================================================================================\n");
}
```


```c
// FUNCTION OF TREE WHICH IS 1ST HURDEL.

void tree(){
    printf("      //\\\\      \n");
    printf("     //  \\\\      \n");
    printf("    //    \\\\    \n");
    printf("   //______\\\\   \n");
    printf("   |        |  \n");
    printf("   |        |  \n");
    printf("   |  TREE  |  \n");
    printf("   |        |  \n");
    printf("   |________|  \n");
    printf("       |||      \n");
    printf("       |||      \n");
    printf("    ___|||___  \n");
    printf("   |   |||   |  \n");
    printf("   | BACKPACK|  \n");
    printf("   |   |||   |  \n");
    printf("   |   |||   |  \n");
    printf("   |_|||_|||_|  \n");
}
```


```c
// FUNCTION OF 3rd GADGET.

void torch(){
    printf(" _______\n");
    printf("|:::::::\\''''''''''\n");
    printf("|:::::::|>''''''''''\n");
    printf("|_______/''''''''''\n");
}
```

3. NOBITA():

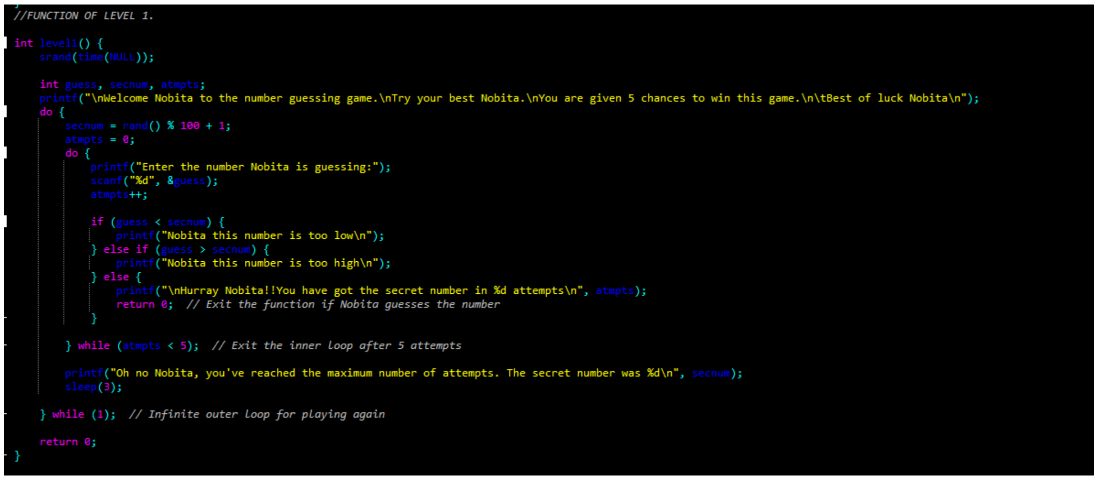
- Displays ASCII art of Nobita, the main character, reinforcing the visual identity of the protagonist.
- Contributes to the overall character immersion in the game.


```c
// FUNCTION OF NOBITA.

void NOBITA() {
    printf("  O\n");
    printf(" /|\\ \n");
    printf(" / \\ \n");
}
```

4. level1():

- Implements the gameplay logic for level 1, where Nobita must guess a secret number.

- Manages the interaction with the player, providing feedback based on their guesses.



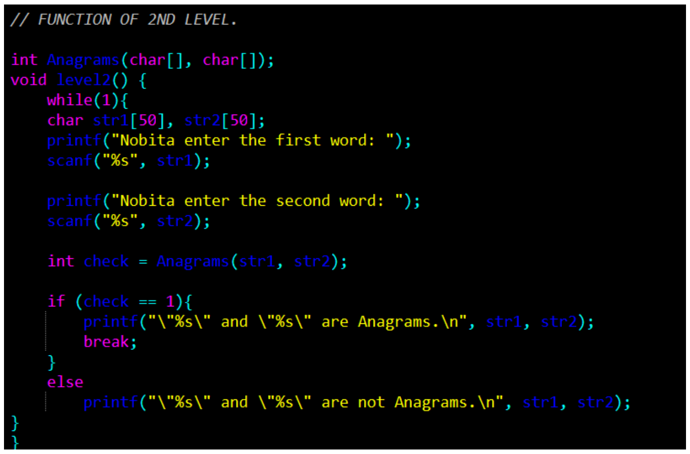
```c
//FUNCTION OF LEVEL 1.

int level1() {
    srand(time(NULL));

    int guess, secnum, atmpts;
    printf("\nWelcome Nobita to the number guessing game.\nTry your best Nobita.\nYou are given 5 chances to win this game.\n\tBest of luck Nobita\n");
    do {
        secnum = rand() % 100 + 1;
        atmpts = 0;
        do {
            printf("Enter the number Nobita is guessing:");
            scanf("%d", &guess);
            atmpts++;

            if (guess < secnum) {
                printf("Nobita this number is too low\n");
            } else if (guess > secnum) {
                printf("Nobita this number is too high\n");
            } else {
                printf("\nHurray Nobita!!You have got the secret number in %d attempts\n", atmpts);
                return 0;  // Exit the function if Nobita guesses the number
            }

        } while (atmpts < 5);  // Exit the inner loop after 5 attempts

        printf("Oh no Nobita, you've reached the maximum number of attempts. The secret number was %d\n", secnum);
        sleep(3);

    } while (1);  // Infinite outer loop for playing again

    return 0;
}
```

5. Anagrams(char[], char[]):

- Determines if two words provided by the player are anagrams.
- Used in the gameplay for level 2, adding a word puzzle challenge.

6. level2():

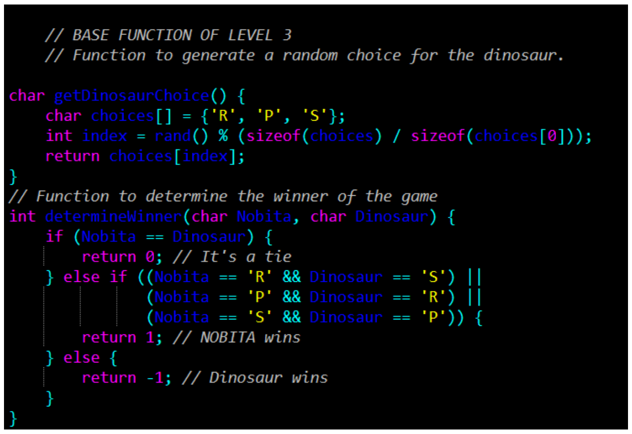
- Implements the gameplay for level 2, involving solving anagrams by inputting pairs of words.
- Manages user input and provides feedback based on the correctness of their answers.



```c
// FUNCTION OF 2ND LEVEL.

int Anagrams(char[], char[]);
void level2() {
    while(1){
    char str1[50], str2[50];
    printf("Nobita enter the first word: ");
    scanf("%s", str1);

    printf("Nobita enter the second word: ");
    scanf("%s", str2);

    int check = Anagrams(str1, str2);

    if (check == 1){
        printf("\"%s\" and \"%s\" are Anagrams.\n", str1, str2);
        break;
    }
    else
        printf("\"%s\" and \"%s\" are not Anagrams.\n", str1, str2);
}
}
```

7. getDinosaurChoice(), determineWinner(char, char):

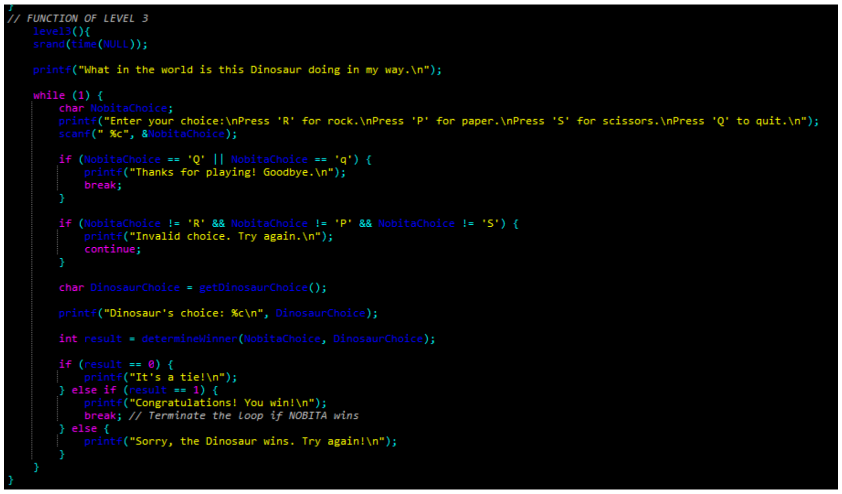
- Functions for the rock-paper-scissors game in level 3.
- getDinosaurChoice() generates a random choice for the dinosaur.
- determineWinner(char Nobita, char Dinosaur) determines the winner based on the choices of Nobita and the dinosaur.



```c
    // BASE FUNCTION OF LEVEL 3
    // Function to generate a random choice for the dinosaur.

char getDinosaurChoice() {
    char choices[] = {'R', 'P', 'S'};
    int index = rand() % (sizeof(choices) / sizeof(choices[0]));
    return choices[index];
}
// Function to determine the winner of the game
int determineWinner(char Nobita, char Dinosaur) {
    if (Nobita == Dinosaur) {
        return 0; // It's a tie
    } else if ((Nobita == 'R' && Dinosaur == 'S') ||
               (Nobita == 'P' && Dinosaur == 'R') ||
               (Nobita == 'S' && Dinosaur == 'P')) {
        return 1; // NOBITA wins
    } else {
        return -1; // Dinosaur wins
    }
}
```

8. level3():

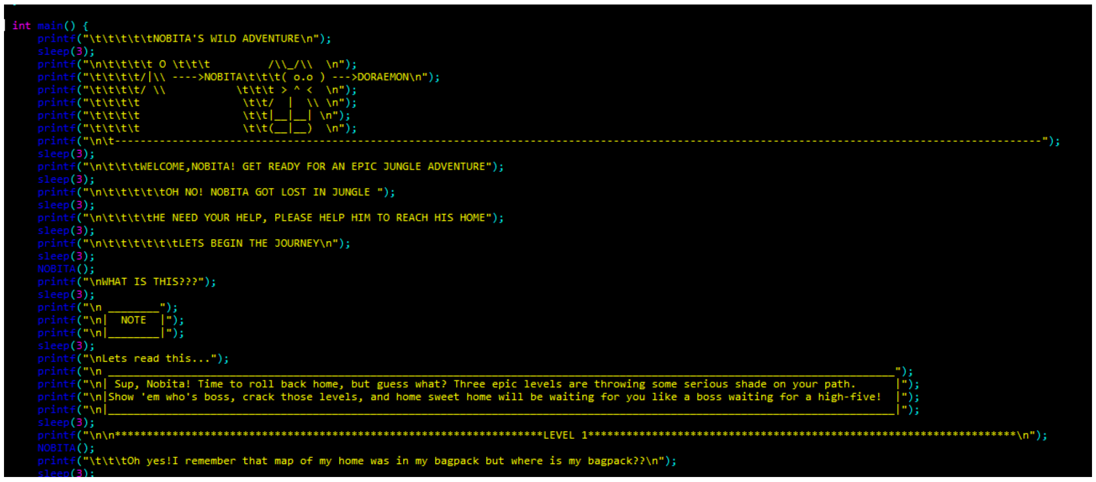
- Implements the gameplay for level 3, where Nobita plays rock-paper-scissors with a dinosaur.
- Manages user input and provides feedback based on the game outcome.



```c
// FUNCTION OF LEVEL 3
    level3(){
    srand(time(NULL));

    printf("What in the world is this Dinosaur doing in my way.\n");

    while (1) {
        char NobitaChoice;
        printf("Enter your choice:\nPress 'R' for rock.\nPress 'P' for paper.\nPress 'S' for scissors.\nPress 'Q' to quit.\n");
        scanf(" %c", &NobitaChoice);

        if (NobitaChoice == 'Q' || NobitaChoice == 'q') {
            printf("Thanks for playing! Goodbye.\n");
            break;
        }

        if (NobitaChoice != 'R' && NobitaChoice != 'P' && NobitaChoice != 'S') {
            printf("Invalid choice. Try again.\n");
            continue;
        }

        char DinosaurChoice = getDinosaurChoice();

        printf("Dinosaur's choice: %c\n", DinosaurChoice);

        int result = determineWinner(NobitaChoice, DinosaurChoice);

        if (result == 0) {
            printf("It's a tie!\n");
        } else if (result == 1) {
            printf("Congratulations! You win!\n");
            break; // Terminate the loop if NOBITA wins
        } else {
            printf("Sorry, the Dinosaur wins. Try again!\n");
        }
    }
}
```

9. main():

- Orchestrates the entire game, presenting the storyline, introducing levels, and calling various functions to implement gameplay.
- Serves as the entry point for the program, coordinating the flow of the game.



```
int main() {
    printf("\t\t\t\t\tNOBITA'S WILD ADVENTURE\n");
    sleep(3);
    printf("\n\t\t\t\t\t O \t\t\t         /\\_/\\  \n");
    printf("\t\t\t\t\t/|\\ ---->NOBITA\t\t\t( o.o ) --->DORAEMON\n");
    printf("\t\t\t\t\t/ \\           \t\t\t > ^ <  \n");
    printf("\t\t\t\t\t                \t\t/  |  \\ \n");
    printf("\t\t\t\t\t                \t\t|__|__| \n");
    printf("\t\t\t\t\t                \t\t(__|__)  \n");
    printf("\n\t-------------------------------------------------------------------------------------------------------------------------------------------------");
    sleep(3);
    printf("\n\t\t\tWELCOME,NOBITA! GET READY FOR AN EPIC JUNGLE ADVENTURE");
    sleep(3);
    printf("\n\t\t\t\t\tOH NO! NOBITA GOT LOST IN JUNGLE ");
    sleep(3);
    printf("\n\t\t\t\tHE NEED YOUR HELP, PLEASE HELP HIM TO REACH HIS HOME");
    sleep(3);
    printf("\n\t\t\t\t\t\tLETS BEGIN THE JOURNEY\n");
    sleep(3);
    NOBITA();
    printf("\nWHAT IS THIS???");
    sleep(3);
    printf("\n ________");
    printf("\n|  NOTE  |");
    printf("\n|________|");
    sleep(3);
    printf("\nLets read this...");
    printf("\n _________________________________________________________________________________________________________________________");
    printf("\n| Sup, Nobita! Time to roll back home, but guess what? Three epic levels are throwing some serious shade on your path.      |");
    printf("\n|Show 'em who's boss, crack those levels, and home sweet home will be waiting for you like a boss waiting for a high-five!  |");
    printf("\n|_________________________________________________________________________________________________________________________|");
    sleep(3);
    printf("\n\n*******************************************************************LEVEL 1********************************************************************\n");
    NOBITA();
    printf("\t\t\tOh yes!I remember that map of my home was in my bagpack but where is my bagpack??\n");
    sleep(3);
```

LIMITATIONS

1. Limited User Interaction:

The game relies heavily on text-based interactions, potentially limiting engagement compared to graphical interfaces.

2. Deterministic Gameplay:

The outcomes of the game are predictable, and repeated plays may follow the same sequence of events, reducing replay ability.

3. Minimal Error Handling:

The provided game code assumes that users will input correct values and lacks comprehensive error-handling mechanisms. This absence of robust error handling increases the risk of encountering unexpected behaviors when users provide invalid inputs or violate expected input formats. Without proper validation and error-checking procedures, the program may exhibit undesired outcomes or even face potential crashes when faced with input data that deviates from the anticipated format or range. To enhance the code's reliability and user experience, it would be advisable to implement more rigorous input validation and error-handling routines.

4. Text-Only Presentation:

The absence of graphical elements and sound effects represents a potential limitation in providing an immersive gaming experience, as it hinders the ability to engage players through visual and auditory stimuli. In modern gaming, the integration of high-quality graphics and dynamic soundscapes contributes significantly to the overall atmosphere, enhancing the sense of realism and emotional connection with the virtual environment. The absence of these elements in a game may result in a less captivating and less interactive user experience, limiting the depth and sensory appeal compared to contemporary gaming standards.

5. Linear Storyline:

The game adheres to a linear storyline, offering a narrative structure with a predetermined sequence of events and limited opportunities for divergent paths or alternate outcomes influenced by player choices. In this linear progression, players follow a pre-established plotline, experiencing the game's events in a fixed order, without significant variations based on their decisions. This approach contrasts with more nonlinear game designs that allow for greater player agency, multiple story branches, and diverse outcomes, providing a more dynamic and personalized gaming experience.

Future ENHANCEMENTS

1. Graphics and Sound Integration:

Introduce graphical elements and sound effects to enhance the visual and auditory experience of the game.

2. Dynamic Storyline:

Implement a branching narrative, allowing player choices to influence the storyline and outcomes.

3. Difficulty Levels:

Introduce adjustable difficulty levels for each challenge, catering to both novice and experienced players.

4. Multiplayer Support:

Enable multiplayer functionality, allowing players to collaborate or compete in the adventure.

5. Extended Gameplay:

Add more levels, challenges, and characters to extend the gameplay and increase overall game depth.

6. User Interface Enhancements:

Develop a graphical user interface (GUI) for a more visually appealing and intuitive gaming experience.

7. Save and Load Functionality:

Implement the ability for players to save their progress and resume the game later.

8. Platform Compatibility:

Adapt the game for various platforms, including desktop, mobile, and web, to reach a broader audience.

CONCLUSION

The overall idea was to develop a user friendly and efficient game. **"NOBITA's Wild Adventure"** provides a fun and challenging experience for players. While the current version has limitations, potential future enhancements can make it even more engaging. The game showcases the creativity and programming skills of the developer, offering an entertaining journey for players willing to help NOBITA reach his home.