BAHRIA UNIVERSITY KARACHI CAMPUS

DEPARTMENT OF COMPUTER SCIENCE

Bahria University
Discovering Knowledge

DATA STRUCTURES AND ALGORITHMS CSL-221

CS 3A

COLOR SORTING GAME

Submitted by:

Alishba Siraj 02-134222-005

Nahin Fatima 02-134222-102

Maha Shahid 02-134222-103

Submitted To:

Engr. Rabia Amjad

Submitted On:

11/1/2024

ACKNOWLEDGEMENT:

We would like to express our sincere appreciation to our educational institute Bahria University for providing us with the foundational knowledge and skills that enabled us to undertake and complete this project. We are grateful for the support and guidance received from our mentors, who played a crucial role in steering us through the complexities of game development. Additionally, we extend our thanks to the programming community for its wealth of resources and open-source projects, which served as valuable references and inspiration during the development process. Collaborating and learning from the broader community has significantly enriched our understanding and implementation of programming concepts. Lastly, we acknowledge the collaborative effort of our three team members who actively contributed to the creation of this Color Sorting Game. Each member's unique skills and dedication played a key role in the successful realization of this project.

ABSTRACT:

Immerse yourself in a vibrant world of color with the Color Sorting Game, a text-based adventure where logic and strategy are your tools! Choose your difficulty, from breezy sorting to mind-bending puzzles, as you face a dynamic list of colored elements begging to be placed in their rightful stacks. With each strategic choice, you earn points and inch closer to victory. This interactive playground of hues is perfect for all ages, sharpening your decision-making skills while keeping you entertained. So, dive into the colorful chaos and let the sorting begin!

TABLE OF CONTENTS PAGE # NO:

INTRODUCTION:

The "Color Sorting Game" is a captivating console-based application developed in C++. This project challenges players to strategically organize a list of colored elements into corresponding color stacks. With three difficulty levels, the game aims to enhance problem-solving skills and offers an engaging experience for users with varying levels of expertise. This report provides insights into the design, implementation, and features of the game, showcasing the effective use of fundamental programming concepts to create an interactive and enjoyable sorting experience.

PROBLEM STATEMENT:

The Color Sorting Game project addresses the need for an entertaining and educational console-based game to enhance users' problem-solving skills and logical reasoning. The goal is to create a challenging and engaging platform where players must efficiently sort a list of colored elements into their respective color stacks. The project offers three difficulty levels to cater to users with different skill levels, providing an enjoyable experience while reinforcing fundamental programming concepts. The challenge lies in implementing the sorting algorithm and user interface effectively, ensuring an interactive and rewarding gaming experience for players.

OBJECTIVES AND GOALS:

Objectives:

Educational Engagement: Develop a game to enhance logical reasoning and problem-solving skills through interactive gameplay.

Programming Proficiency: Provide a project that allows users to apply and improve their programming skills, particularly in C++.

Difficulty Levels: Cater to users with varying skill levels by incorporating three difficulty levels (Easy, Intermediate, Difficult).

User Interface Design: Create an intuitive and user-friendly interface for a seamless gaming experience.

Goals:

Clear Instructions: Implement clear instructions, visually appealing displays, and easy-to-understand prompts for an enjoyable user experience.

Randomization and Shuffling: Utilize randomization to dynamically generate colored elements, ensuring a unique experience in each gameplay session.

Efficient Sorting Algorithm: Implement a robust sorting algorithm that responds to user choices, providing a challenging yet solvable puzzle.

Scoring System: Integrate a scoring system that reflects users' performance in sorting elements, offering feedback and motivation for improvement.

Game Completion Check: Include a mechanism to acknowledge the successful completion of the game, congratulating players on their achievement.

PROJECT SCOPE:

The scope of the Color Sorting Game project encompasses the development of an interactive and educational game designed to enhance logical reasoning and programming skills. The primary focus is on creating a challenging yet entertaining experience for users, allowing them to engage with the game at various difficulty levels. The project includes the following key components.

Gameplay Mechanism:

Users will interact with a list of colored elements represented by letters (R, G, B) and sort them into their respective color stacks.

Three difficulty levels (Easy, Intermediate, Difficult) will cater to users with different skill levels.

User Interface:

A visually appealing and user-friendly interface will be designed to provide clear instructions, game status, and stack displays.

Instructions will guide users on sorting elements and making choices based on the selected difficulty level.

Randomization and Shuffling:

The game will dynamically generate colored elements using randomization techniques, ensuring a unique set of elements for each gameplay session.

Elements will be shuffled to create challenging scenarios for users to solve.

Sorting Algorithm:

An efficient sorting algorithm will be implemented to respond to user choices (color selection) and manage the sorting process.

The algorithm will enforce the rules of the game and validate user inputs.

Scoring System:

A scoring system will be integrated to evaluate users' performance in sorting elements.

Scores will be calculated based on the number of correctly sorted elements, providing feedback to users.

Game Completion:

The game will include a mechanism to recognize and acknowledge the successful completion of the sorting task.

Users will receive congratulatory messages upon successfully sorting all elements.

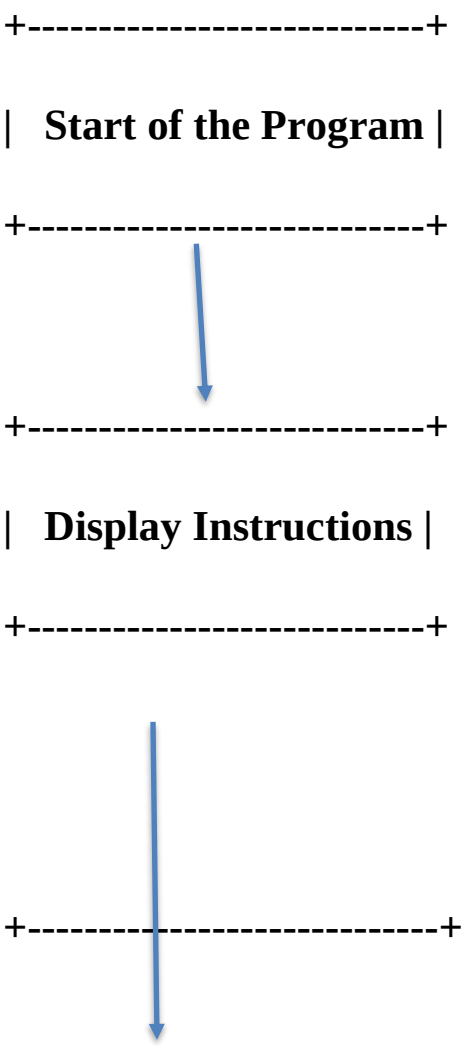
Error Handling:

The system will incorporate error handling mechanisms to guide users in case of invalid inputs or exceeding maximum move limits.

Educational Component:

The game aims to serve as an educational tool, fostering logical thinking and problem-solving skills among users.

WORKFLOW:



| **Choose Level (1/2/3)** |

+----------------------------+

+----------------------------------+

| **Initialize Game Elements** |

+----------------------------------+

+--------------------------------------+

| **While (Top Index < Max Size)** |

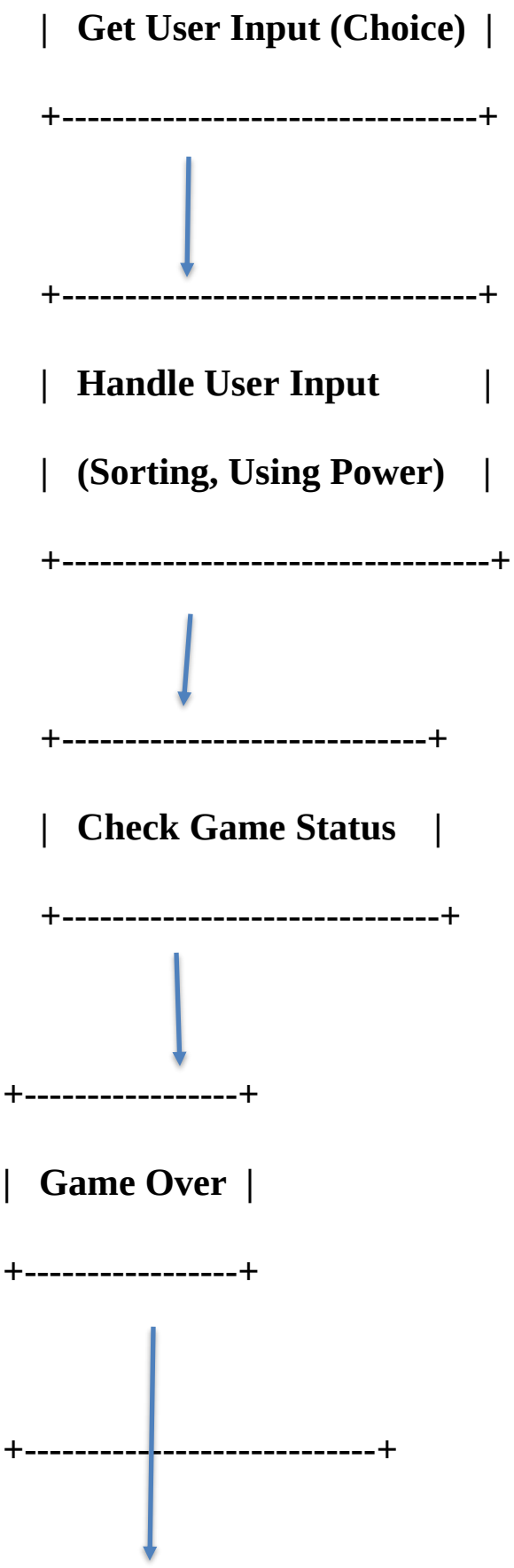
+--------------------------------------+

+----------------------------+

| **Display List & Stacks** |

+----------------------------+

+------------------------------+


| Get User Input (Choice) |
+-----------------------------+
+-----------------------------+
| Handle User Input |
| (Sorting, Using Power) |
+-----------------------------+
+---------------------------+
| Check Game Status |
+---------------------------+
+----------------+
| Game Over |
+----------------+
+-------------------------+

| **Display Final Score** |

+-------------------------+

+-----------------------+

| **Continue / Exit (0)** |

+-----------------------+

+--------------------------+

| **End of the Program** |

+--------------------------+

OVERVIEW OF PROJECT:

The Color Sorting Game is a console-based project implemented in C++. It presents players with a list of colored elements (R, G, B) that need to be sorted into their corresponding color stacks. The game offers three difficulty levels to cater to different skill levels. Users input the correct color letter to sort elements into the respective stacks. The goal is to complete the sorting process efficiently within the given constraints. The project provides an engaging and educational platform, reinforcing problem-solving skills and logical reasoning.

TOOLS AND TECHNOLOGIES:

The Color Sorting Game is developed using the C++ programming language, making use of the Standard Template Library (STL) for efficient implementation of data structures and algorithms. The game is presented through a console-based interface, where users interact with the program through basic input and output operations. Random number generation is a key aspect of the game, and it is achieved using the rand() function in C++. This project employs a minimalistic yet effective set of tools and technologies, focusing on simplicity and accessibility for users interacting with the game through the console environment.

PROJECT FEATURES:

The Color Sorting Game project includes the following features:

Interactive Console Interface:

The game provides an interactive console-based interface for users to play and interact with the color sorting challenge.

Dynamic Instructions:

The game begins with dynamic instructions, providing the user with information on how to play and achieve the goal.

Three Difficulty Levels:

Users can choose from three difficulty levels: Easy, Medium, and Difficult. Each level introduces different challenges in sorting colored elements.

Random Element Generation:

The game generates a sequence of colored elements randomly based on the chosen difficulty level. This adds variability and challenge to each playthrough.

Multiple Stacks for Sorting:

Depending on the difficulty level, users may have to sort elements into different color stacks. The number of stacks varies based on the chosen level.

Scoring System:

A scoring system is implemented to track the player's performance. Players are awarded points for correctly sorting elements.

Game Progress Tracking:

The game keeps track of the player's progress, including the number of moves made, the remaining moves allowed, and the status of sorted elements.

User Input Validation:

The program validates user input to ensure that only valid choices are accepted, enhancing the reliability of the game.

Game Over Conditions:

The game can end in different ways, such as when the user completes the sorting task successfully, chooses to quit, exceeds the maximum allowed moves, or makes an invalid move.

Congratulations Message:

Upon successfully sorting all elements, the player receives a congratulatory message, and the total score is displayed.

Dynamic Level Configuration:

Each difficulty level has different configurations, such as the maximum number of moves allowed, the number of color stacks, and the types of colors in the sequence.

Flexible Stack Choices:

In Difficult mode, users have four stack choices (Stack1, Stack2, Stack3, Stack4) for sorting elements, providing a more challenging sorting experience.

DATA STRUCTURES IMPLEMENTED CONCEPTS:

Here are the key data structures and concepts utilized:

Arrays:

Arrays are extensively used to represent various components of the game, such as the sequence of colored elements, different color stacks, and counts of colors.

Stacks:

The game utilizes stacks to implement the sorting mechanism. Different color stacks (Red, Green, Blue, Stack1, Stack2, Stack3, Stack4) are implemented using arrays to keep track of sorted elements.

Randomization:

The project incorporates randomization to generate a sequence of colored elements in the Intermediate and Difficult levels. This randomness is achieved using a custom random number generation function.

Seed for Randomization:

The seed value is used to initialize the random number generator, ensuring different random sequences in each game session.

Switch Statements:

Switch statements are employed for handling user input and directing the program's flow based on the chosen difficulty level and the user's sorting decisions.

CODE:

```
#include <iostream>

#include <ctime>

#include <stack>


#define MAX_SIZE 15

#define STACK_SIZE 5


using namespace std;


void displayInstructions()

{

   cout << "\t\t\t\t\t ----------------------------------------------------" <<
endl;

   cout << "\t\t\t\t\t|          Welcome to the Color Sorting Game           |"
<< endl;

   cout << "\t\t\t\t\t|                                                      |" << endl;
```

```
    cout << "\t\t\t\t\t ----------------------------------------------------" <<
endl;

    cout << endl << endl << endl;

    cout << "\t\t\t*INSTRUCTIONS*" << endl;

    cout << endl;

    cout << "1) ---> You will see a list of colored elements (R, G, B)." <<
endl;

    cout << "2) ---> Your goal is to sort these elements into their
respective color stacks." << endl;

    cout << "3) ---> The elements are represented in the list with (R, G,
B)." << endl;

    cout << "4) ---> Enter the color letter to sort from the list into its
corresponding stack." << endl;

    cout << "5) ---> For example, if 'R' is at the top, enter 'R' to sort it into
the Red stack." << endl;

    cout << "6) ---> Keep sorting until all elements are in their correct
stacks to win." << endl << endl;

    cout << "7) ---> In Level 2 (for >=30 points) and 3 (for >=55 points) a
special power is unlocked known as PowerStack." << endl << endl;
```

```
    cout << "8) ---> It is an extra stack to store your color but to use it in level 2 your 15 point will be deducted and in level 3, 20 points." << endl << endl;

    cout << "9) ---> Also in level 1 & 2 you have limited moves (16) so be carefull." << endl << endl;

   cout << endl;

    cout << "\t\t\t\t\t ------- GIVE IT A TRY; BEST OF LUCK :) --------" << endl << endl << endl;

}


void displayList(const char elements[], int size, int topIndex)

{

    cout << "List of Elements to Sort: ";

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

    {

        if (i == topIndex)

        {

            cout << "(" << elements[i] << ") ";
```

```
        }
        else
        {
            cout << elements[i] << " ";
        }
    }
    cout << endl;
}

void displayStack(const char stack[], int top, const char* stackName)
{
    cout << stackName << " Stack: ";
    if (top == -1)
    {
        cout << "Empty";
    }
    else
```

```
    {
        for (int i = top; i >= 0; --i)
        {
            cout << stack[i] << " ";
        }
    }
    cout << endl;
}

char getColorToSort(const char elements[], int topIndex)
{
    return elements[topIndex];
}

bool isGameWon(const char redStack[], const char greenStack[], const
char blueStack[], int chosenLevel)
{
    bool isRedSorted = true;
```

```
bool isGreenSorted = true;
bool isBlueSorted = true;

// Check if the Red stack is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (redStack[i] != 'R' || redStack[i] > redStack[i + 1])
    {
        isRedSorted = false;
        break;
    }
}

// Check if the Green stack is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (greenStack[i] != 'G' || greenStack[i] > greenStack[i + 1])
```

```
    {
        isGreenSorted = false;
        break;
    }
}

// Check if the Blue stack is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (blueStack[i] != 'B' || blueStack[i] > blueStack[i + 1])
    {
        isBlueSorted = false;
        break;
    }
}

// Game is won if all three stacks are sorted
```

```
    if (chosenLevel == 2)
    {
        return isRedSorted && isGreenSorted && isBlueSorted;
    }
    else
    {
        return isRedSorted || isGreenSorted || isBlueSorted;
    }
}

bool isGameWon(const char Stack1[], const char Stack2[], const char Stack3[], const char Stack4[])
{
    bool isStack1Sorted = true;
    bool isStack2Sorted = true;
    bool isStack3Sorted = true;
    bool isStack4Sorted = true;
```

```
// Check if Stack1 is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (Stack1[i] != ' ' && (Stack1[i] != Stack1[i + 1]))
    {
        isStack1Sorted = false;
        break;
    }
}

// Check if Stack2 is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (Stack2[i] != ' ' && (Stack2[i] != Stack2[i + 1]))
    {
        isStack2Sorted = false;
        break;
```

```
    }
}

// Check if Stack3 is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (Stack3[i] != ' ' && (Stack3[i] != Stack3[i + 1]))
    {
        isStack3Sorted = false;
        break;
    }
}

// Check if Stack4 is sorted
for (int i = 0; i < STACK_SIZE - 1; ++i)
{
    if (Stack4[i] != ' ' && (Stack4[i] != Stack4[i + 1]))
```

```
        {
            isStack4Sorted = false;
            break;
        }
    }

    // Game is won if all stacks are sorted
    return isStack1Sorted && isStack2Sorted && isStack3Sorted &&
isStack4Sorted;
}


unsigned int generateRandomSeed()
{
    return static_cast<unsigned int>(time(nullptr));
}

int myRandom(unsigned int& seed, int min, int max)
```

```
{
    seed = seed * 1103515245 + 12345;
    return static_cast<int>((seed / 65536) % (max - min + 1)) + min;
}



void initializeElements(char elements[], unsigned int& seed, int
chosenLevel)
{
    if (chosenLevel == 1)
    {
        for (int i = 0; i < MAX_SIZE; i++)
        {
            elements[i] = (i % 3 == 0) ? 'R' : ((i % 3 == 1) ? 'G' : 'B');
        }
    }
    else if (chosenLevel == 2)
    {
```

```
        for (int i = 0; i < MAX_SIZE; i++)
        {
            elements[i] = myRandom(seed, 0, 2) == 0 ? 'R' :
(myRandom(seed, 0, 1) == 1 ? 'G' : 'B');
        }
    }
    else // Level 3 with 4 colors (R, G, B, Y) and 4 stacks (Stack1, Stack2,
Stack3, Stack4)
    {
        for (int i = 0; i < MAX_SIZE; i++)
        {
            int randomColor = myRandom(seed, 0, 3);
            if (randomColor == 0)
            {
                elements[i] = 'R';
            }
            else if (randomColor == 1)
            {
```

```
            elements[i] = 'G';
        }
        else if (randomColor == 2)
        {
            elements[i] = 'B';
        }
        else
        {
            elements[i] = 'Y';
        }
    }
  }
}
void handleFullStack(char colorToSort, char powerStack[], int&
powerTop, int& score, int chosenLevel)
{
  if (chosenLevel == 2)
  {
```

```
        cout << "\t\t\t\t\t" << colorToSort << " Stack Full! Using Power Stack." << endl;
        if (powerTop < STACK_SIZE - 1)
        {
            powerStack[++powerTop] = colorToSort;
            score = score -15; // Deduct 15 points for using the power stack
        }
        else
        {
            cout << "\t\t\t\t\tPower Stack Full! Cannot use it now." << endl;
        }
    }
    else if (chosenLevel == 3)
    {
        cout << "\t\t\t\t\t" << colorToSort << " Stack Full! Using Power Stack." << endl;
        if (powerTop < STACK_SIZE - 1)
        {
```

```
            powerStack[++powerTop] = colorToSort;
            score = score - 20; // Deduct 20 points for using the power stack
        }
        else
        {
            cout << "\t\t\t\t\tPower Stack Full! Cannot use it now." << endl;
        }
    }
}

int main()
{
    int moves = 0; // Initialize move counter
    unsigned int seed = generateRandomSeed();

    char elements[MAX_SIZE];
    int topIndex = 0; // Top index of the element to sort
```

```
    int opt = 1;
    char redStack[STACK_SIZE], greenStack[STACK_SIZE],
blueStack[STACK_SIZE], powerStack[MAX_SIZE];
    int redTop = -1, greenTop = -1, blueTop = -1, powerTop = -1;

    char Stack1[STACK_SIZE], Stack2[STACK_SIZE],
Stack3[STACK_SIZE], Stack4[STACK_SIZE];
    int stack1Top = -1, stack2Top = -1, stack3Top = -1, stack4Top = -1;

    int score = 0;
    int chosenLevel;
    while(opt!=0)
    {
    displayInstructions();

    cout << "Choose the type of level to play:" << endl;
    cout << "1. Easy" << endl;
    cout << "2. Medium" << endl;
```

```
    cout << "3. Difficult" << endl;
    cout << "Enter your choice (1/2/3): ";
    cin >> chosenLevel;
    cout << endl;

    if (chosenLevel != 1 && chosenLevel != 2 && chosenLevel != 3)
    {
        cout << "\t\t\t-- INVALID CHOICE!!! Exiting the game... --" << endl;
        cout << "\t\t\t\t -------------------------------------" << endl;
        cout << "\t\t\t\t|  Total Points:                  " << score << " | " << endl;
        cout << "\t\t\t\t -------------------------------------" << endl;
        return 1;
    }

    initializeElements(elements, seed, chosenLevel);
```

```
    char userChoice;

    while (topIndex < MAX_SIZE)
    {
        char colorToSort = getColorToSort(elements, topIndex);

        cout << "Sort the '" << colorToSort << "' from the list." << endl << endl;
        displayList(elements, MAX_SIZE, topIndex);

        if (chosenLevel == 3)
        {
            cout << "Stack Options: " << endl;
            displayStack(Stack1, stack1Top, "Stack 1");
            displayStack(Stack2, stack2Top, "Stack 2");
            displayStack(Stack3, stack3Top, "Stack 3");
            displayStack(Stack4, stack4Top, "Stack 4");
```

```
    displayStack(powerStack, powerTop, "Power");


}
else
{
    displayStack(redStack, redTop, "Red");
    displayStack(greenStack, greenTop, "Green");
    displayStack(blueStack, blueTop, "Blue");
    displayStack(powerStack, powerTop, "Power");
}

cout << endl;




if (chosenLevel == 1 || chosenLevel == 2)
{
```

```
        cout << "Enter your choice (R/G/B/P for Power) or 'Q' to quit: ";
        cin >> userChoice;

        if (userChoice == 'Q' || userChoice == 'q')
        {
            cout << "Exiting the game." << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            cout << "\t\t\t\t|  Total Points:                  " << score << " | "
<< endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            break;
        }
        if (userChoice == 'R' || userChoice == 'G' || userChoice == 'B')
        {
            if (userChoice == colorToSort)
            {
                if (userChoice == 'R' && redTop < STACK_SIZE - 1)
                {
```

```
            redStack[++redTop] = 'R';
        }
        else if (userChoice == 'G' && greenTop < STACK_SIZE -
1)
        {
            greenStack[++greenTop] = 'G';
        }
        else if (userChoice == 'B' && blueTop < STACK_SIZE - 1)
        {
            blueStack[++blueTop] = 'B';
        }
        else if (powerTop < STACK_SIZE - 1)
        {
            handleFullStack(userChoice, powerStack, powerTop,
score, chosenLevel);
        }
        else
        {
```

```
                handleFullStack(userChoice, powerStack, powerTop,
score, chosenLevel);
            }


            elements[topIndex++] = ' '; // Replace the sorted element
with a placeholder
            score += 5; // Increment score for correct sorting
        }
        else
        {
            cout << endl;
            cout << "\t\t\t\t\tINVALID CHOICE!!!" << endl;
        }
    }


    else if (userChoice == 'P' || userChoice == 'p')
    {
        if (score >= 30 && powerTop < STACK_SIZE - 1)
```

```
            {
                powerStack[++powerTop] = colorToSort; // Allow putting
the same color into the power stack
                score -= 15; // Deduct 15 points for using the power stack
                cout << "You used power stack special power." << endl;
            }
            else if (powerTop == STACK_SIZE - 1)
            {
                cout << "\t\t\t\t\tPower Stack Full! Cannot use it now." <<
endl;
            }
            else
            {
                cout << "\t\t\t\t\tInsufficient points to use Power Stack." <<
endl;
            }
        }
        else
```

```
        {
            cout << endl;
            cout << "\t\t\t\t\tINVALID CHOICE!!!" << endl;
        }
        moves++; // Increment move counter

        if (moves > MAX_SIZE + 1) {
            cout << "Out of moves! You've exceeded the maximum moves allowed." << endl;
            cout << "Game Over!" << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            cout << "\t\t\t\t|  Total Points:                  " << score << " | " << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            break;
        }
    }
    if (chosenLevel == 3)
```

```
    {
        int stackChoice;
        cout << "Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': ";
        cin >> stackChoice;

        if (stackChoice == 5)
        {
            cout << "Game Over!" << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            cout << "\t\t\t\t|  Total Points:                  " << score << " | " << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            break;
        }

        if (stackChoice >= 1 && stackChoice <= 4)
        {
```

```
char userChoice = getColorToSort(elements, topIndex);


switch (stackChoice)
{
case 1:
   if (userChoice == 'R' && stack1Top < STACK_SIZE - 1)
   {
      Stack1[++stack1Top] = 'R';
   }
   else if (userChoice == 'G' && stack1Top < STACK_SIZE -
1)
   {
      Stack1[++stack1Top] = 'G';
   }
   else if (userChoice == 'B' && stack1Top < STACK_SIZE -
1)
   {
      Stack1[++stack1Top] = 'B';
```

```
            }
            else if (userChoice == 'Y' && stack1Top < STACK_SIZE -
1)
            {
                Stack1[++stack1Top] = 'Y';
            }
            else
            {
                handleFullStack(userChoice, powerStack, powerTop,
score, chosenLevel);
            }
            break;
        case 2:
            if (userChoice == 'R' && stack2Top < STACK_SIZE - 1)
            {
                Stack2[++stack2Top] = 'R';
            }
```

```
        else if (userChoice == 'G' && stack2Top < STACK_SIZE -
1)
        {
           Stack2[++stack2Top] = 'G';
        }
        else if (userChoice == 'B' && stack2Top < STACK_SIZE -
1)
        {
           Stack2[++stack2Top] = 'B';
        }
        else if (userChoice == 'Y' && stack2Top < STACK_SIZE -
1)
        {
           Stack2[++stack2Top] = 'Y';
        }
        else
        {
```

```
                handleFullStack(userChoice, powerStack, powerTop,
score, chosenLevel);
            }
            break;
        case 3:
            if (userChoice == 'R' && stack3Top < STACK_SIZE - 1)
            {
                Stack3[++stack3Top] = 'R';
            }
            else if (userChoice == 'G' && stack3Top < STACK_SIZE -
1)
            {
                Stack3[++stack3Top] = 'G';
            }
            else if (userChoice == 'B' && stack3Top < STACK_SIZE -
1)
            {
                Stack3[++stack3Top] = 'B';
```

```
            }
            else if (userChoice == 'Y' && stack3Top < STACK_SIZE -
1)
            {
                Stack3[++stack3Top] = 'Y';
            }
            else
            {
                handleFullStack(userChoice, powerStack, powerTop,
score, chosenLevel);
            }
            break;
        case 4:
            if (userChoice == 'R' && stack4Top < STACK_SIZE - 1)
            {
                Stack4[++stack4Top] = 'R';
            }
```

```
        else if (userChoice == 'G' && stack4Top < STACK_SIZE -
1)
        {
           Stack4[++stack4Top] = 'G';
        }
        else if (userChoice == 'B' && stack4Top < STACK_SIZE -
1)
        {
           Stack4[++stack4Top] = 'B';
        }
        else if (userChoice == 'Y' && stack4Top < STACK_SIZE -
1)
        {
           Stack4[++stack4Top] = 'Y';
        }
        else
        {
```

```
                handleFullStack(userChoice, powerStack, powerTop,
score, chosenLevel);
                }
                break;
            default:
                cout << "Invalid choice or stack is full!" << endl;
                continue; // Continue loop without incrementing topIndex
            }

            elements[topIndex++] = ' '; // Mark the element as sorted
            score += 5; // Increment score for correct sorting
        }

        else if (stackChoice == 6)
        {
            if (score >= 55 && powerTop < STACK_SIZE - 1)
            {
```

```
                powerStack[++powerTop] = colorToSort; // Allow putting
the same color into the power stack
                score -= 20; // Deduct 20 points for using the power stack
            }
            else if (powerTop == STACK_SIZE - 1)
            {
                cout << "\t\t\t\t\tPower Stack Full! Cannot use it now." <<
endl;
            }
            else
            {
                cout << "\t\t\t\t\tInsufficient points to use Power Stack." <<
endl;
            }
        }
        else
        {
            cout << endl;
```

```
            cout << "\t\t\t\t\tINVALID CHOICE!!!" << endl;
        }



    }



    if (topIndex == MAX_SIZE)
    {
        if (chosenLevel == 1 || chosenLevel == 2)
        {
            if (isGameWon(redStack, greenStack, blueStack,
chosenLevel))
            {
                // Display all sorted elements in the respective stack
                displayStack(redStack, redTop, "Red");
                displayStack(greenStack, greenTop, "Green");
                displayStack(blueStack, blueTop, "Blue");
```

```
            displayStack(powerStack, powerTop, "Power");


            cout << endl << endl;
            cout << "\t\tCONGRATULATIONS!!! All elements are sorted!" << endl << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            cout << "\t\t\t\t|  Total Points:                " << score << " | " << endl;
            cout << "\t\t\t\t ------------------------------------" << endl;
            break;
        }
    }
    else if (chosenLevel == 3)
    {
        if (isGameWon(Stack1, Stack2, Stack3, Stack4))
        {
            // Display sorted stacks and winning message for Level 3
            displayStack(Stack1, stack1Top, "Stack 1");
```

```
                displayStack(Stack2, stack2Top, "Stack 2");
                displayStack(Stack3, stack3Top, "Stack 3");
                displayStack(Stack4, stack4Top, "Stack 4");
                displayStack(powerStack, powerTop, "Power");

                cout << endl << endl;
                cout << "\t\tCONGRATULATIONS!!! All elements are sorted!" << endl << endl;
                cout << "\t\t\t\t ------------------------------------" << endl;
                cout << "\t\t\t\t|  Total Points:                " << score << " | " << endl;
                cout << "\t\t\t\t ------------------------------------" << endl;
                break;
            }
        }
    }
}
```

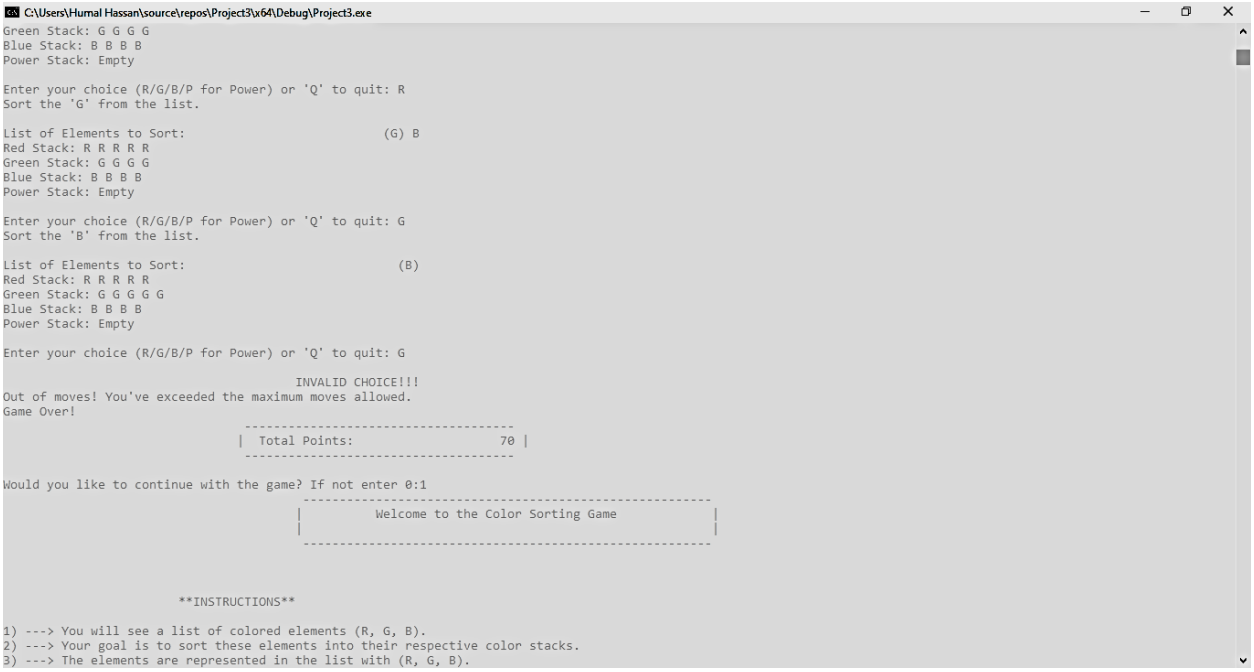
```
    if (chosenLevel == 1 || chosenLevel == 2)
    {
        if (topIndex == MAX_SIZE && !isGameWon(redStack,
greenStack, blueStack, chosenLevel))
        {
            cout << "\n\n\t\t\tAll Sorted! " << endl;
            cout << "\t\t\t\t -------------------------------------" << endl;
            cout << "\t\t\t\t| YOUR FINAL SCORE:               " << score <<
" | " << endl;
            cout << "\t\t\t\t -------------------------------------" << endl;
        }
    }
    else if (chosenLevel == 3)
    {
        if (topIndex == MAX_SIZE && !isGameWon(Stack1, Stack2,
Stack3, Stack4))
        {
            cout << "\n\n\t\t\tAll Sorted! " << endl;
```

```
            cout << "\t\t\t\t -------------------------------------" << endl;
            cout << "\t\t\t\t| YOUR FINAL SCORE:                " << score <<
" | " << endl;
            cout << "\t\t\t\t -------------------------------------" << endl;
        }
    }
    cout << "\nWould you like to continue with the game? If not enter 0:";
    cin >> opt;
}
    cout << "\nThank you for playing :D";

    return 0;
}
```

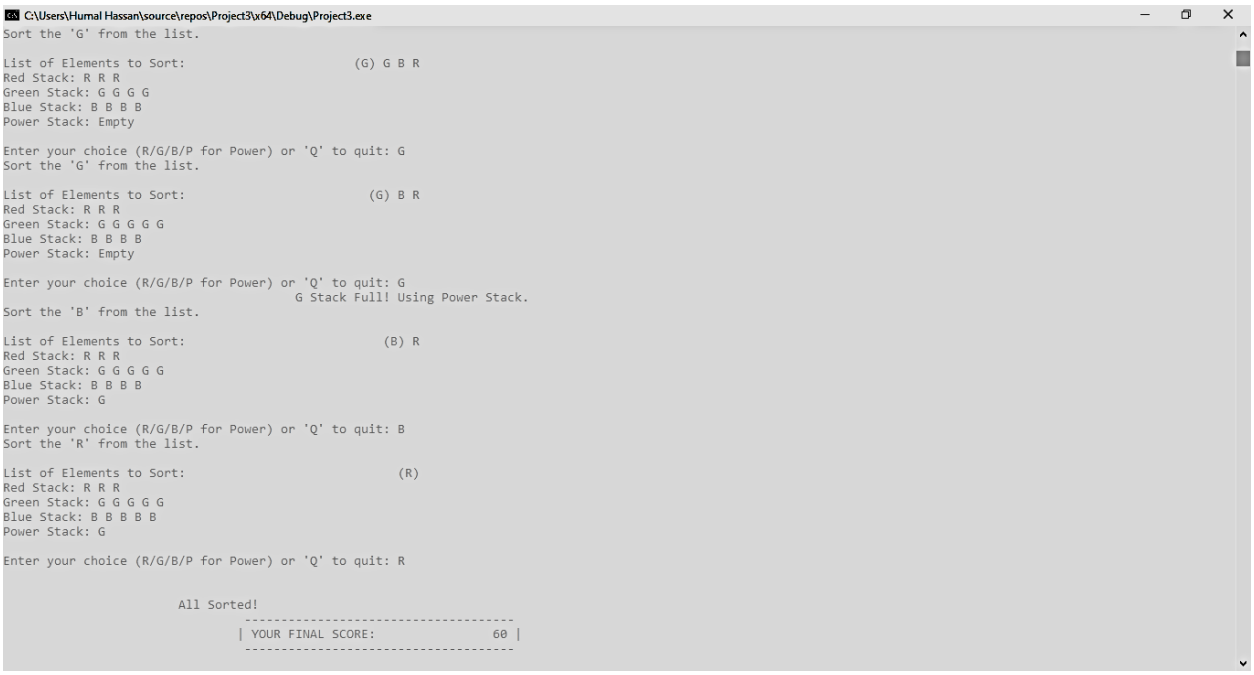
OUTPUT:

```
C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe
 ---------------------------------------------------------
|          Welcome to the Color Sorting Game              |
|                                                         |
 ---------------------------------------------------------



                    **INSTRUCTIONS**

1) ---> You will see a list of colored elements (R, G, B).
2) ---> Your goal is to sort these elements into their respective color stacks.
3) ---> The elements are represented in the list with (R, G, B).
4) ---> Enter the color letter to sort from the list into its corresponding stack.
5) ---> For example, if 'R' is at the top, enter 'R' to sort it into the Red stack.
6) ---> Keep sorting until all elements are in their correct stacks to win.

7) ---> In Level 2 (for >=30 points) and 3 (for >=55 points) a special power is unlocked known as PowerStack.

8) ---> It is an extra stack to store your color but to use it in level 2 your 15 point will be deducted and in level 3, 20 points.

9) ---> Also in level 1 & 2 you have limited moves (16) so be carefull.


                                    ------- GIVE IT A TRY; BEST OF LUCK :) --------


Choose the type of level to play:
1. Easy
2. Medium
3. Difficult
Enter your choice (1/2/3): 1

Sort the 'R' from the list.

List of Elements to Sort: (R) G B R G B R G B R G B R G B
Red Stack: Empty
Green Stack: Empty
Blue Stack: Empty
Power Stack: Empty

Enter your choice (R/G/B/P for Power) or 'Q' to quit: R
Sort the 'G' from the list.

List of Elements to Sort:   (G) B R G B R G B R G B R G B
```

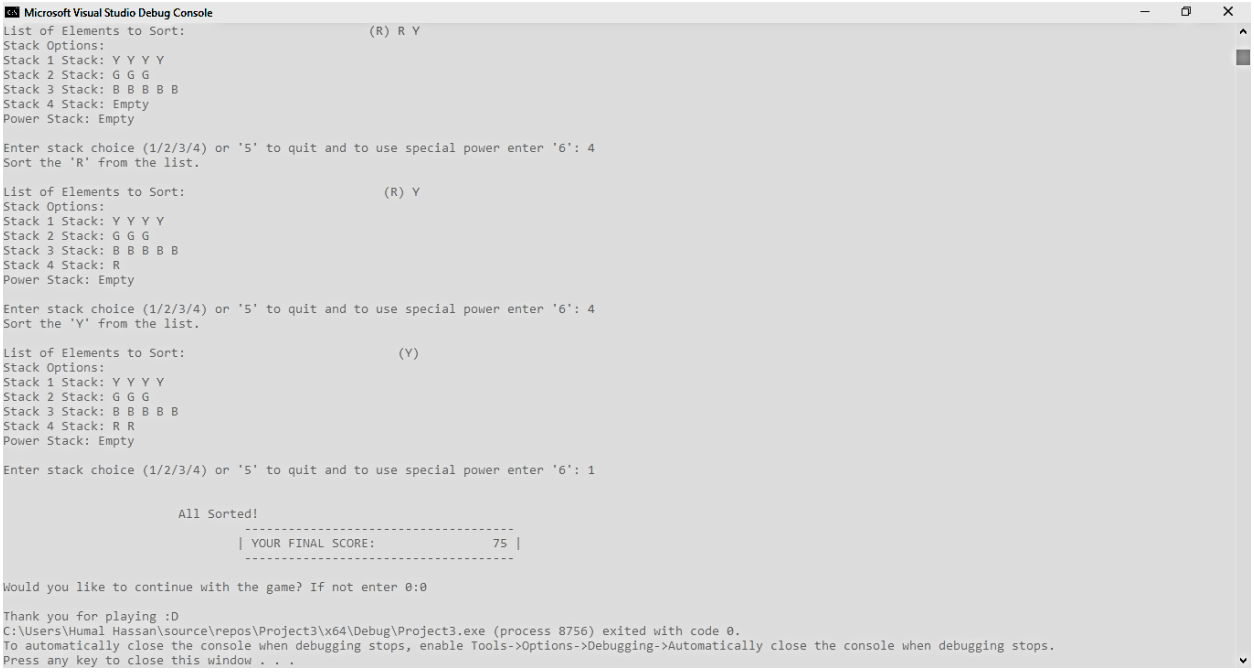
```
C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe

Enter your choice (R/G/B/P for Power) or 'Q' to quit: R
Sort the 'G' from the list.

List of Elements to Sort:                   (G) B R G B
Red Stack: R R R R
Green Stack: G G G
Blue Stack: B B B
Power Stack: Empty

Enter your choice (R/G/B/P for Power) or 'Q' to quit: G
Sort the 'B' from the list.

List of Elements to Sort:                     (B) R G B
Red Stack: R R R R
Green Stack: G G G G
Blue Stack: B B B
Power Stack: Empty

Enter your choice (R/G/B/P for Power) or 'Q' to quit: R

                                  INVALID CHOICE!!!
Sort the 'B' from the list.

List of Elements to Sort:                     (B) R G B
Red Stack: R R R R
Green Stack: G G G G
Blue Stack: B B B
Power Stack: Empty

Enter your choice (R/G/B/P for Power) or 'Q' to quit: B
Sort the 'R' from the list.

List of Elements to Sort:                       (R) G B
Red Stack: R R R R
Green Stack: G G G G
Blue Stack: B B B B
Power Stack: Empty

Enter your choice (R/G/B/P for Power) or 'Q' to quit: G

                                  INVALID CHOICE!!!
Sort the 'R' from the list.
```


C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe
Green Stack: G G G G
Blue Stack: B B B B
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: R
Sort the 'G' from the list.
List of Elements to Sort: (G) B
Red Stack: R R R R R
Green Stack: G G G G
Blue Stack: B B B B
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: G
Sort the 'B' from the list.
List of Elements to Sort: (B)
Red Stack: R R R R R
Green Stack: G G G G G
Blue Stack: B B B B
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: G
INVALID CHOICE!!!
Out of moves! You've exceeded the maximum moves allowed.
Game Over!
| Total Points: 70 |
Would you like to continue with the game? If not enter 0:1
Welcome to the Color Sorting Game
INSTRUCTIONS
1) ---> You will see a list of colored elements (R, G, B).
2) ---> Your goal is to sort these elements into their respective color stacks.
3) ---> The elements are represented in the list with (R, G, B).


C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe
3. Difficult
Enter your choice (1/2/3): 2
Sort the 'G' from the list.
List of Elements to Sort: (G) R B B R R G G G B B G G B R
Red Stack: Empty
Green Stack: Empty
Blue Stack: Empty
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: G
Sort the 'R' from the list.
List of Elements to Sort: (R) B B R R G G G B B G G B R
Red Stack: Empty
Green Stack: G
Blue Stack: Empty
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: R
Sort the 'B' from the list.
List of Elements to Sort: (B) B R R G G G B B G G B R
Red Stack: R
Green Stack: G
Blue Stack: Empty
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: B
Sort the 'B' from the list.
List of Elements to Sort: (B) R R G G G B B G G B R
Red Stack: R
Green Stack: G
Blue Stack: B
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: B
Sort the 'R' from the list.
List of Elements to Sort: (R) R G G G B B G G B R
Red Stack: R
Green Stack: G


C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe
Sort the 'G' from the list.
List of Elements to Sort: (G) G B R
Red Stack: R R R
Green Stack: G G G G
Blue Stack: B B B B
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: G
Sort the 'G' from the list.
List of Elements to Sort: (G) B R
Red Stack: R R R
Green Stack: G G G G G
Blue Stack: B B B B
Power Stack: Empty
Enter your choice (R/G/B/P for Power) or 'Q' to quit: G
G Stack Full! Using Power Stack.
Sort the 'B' from the list.
List of Elements to Sort: (B) R
Red Stack: R R R
Green Stack: G G G G G
Blue Stack: B B B B
Power Stack: G
Enter your choice (R/G/B/P for Power) or 'Q' to quit: B
Sort the 'R' from the list.
List of Elements to Sort: (R)
Red Stack: R R R
Green Stack: G G G G G
Blue Stack: B B B B B
Power Stack: G
Enter your choice (R/G/B/P for Power) or 'Q' to quit: R
All Sorted!
| YOUR FINAL SCORE: 60 |


C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe
Choose the type of level to play:
1. Easy
2. Medium
3. Difficult
Enter your choice (1/2/3): 3
Sort the 'Y' from the list.
List of Elements to Sort: (Y) G B B G Y B G Y B Y B R R Y
Stack Options:
Stack 1 Stack: Empty
Stack 2 Stack: Empty
Stack 3 Stack: Empty
Stack 4 Stack: Empty
Power Stack: Empty
Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': 1
Sort the 'G' from the list.
List of Elements to Sort: (G) B B G Y B G Y B Y B R R Y
Stack Options:
Stack 1 Stack: Y
Stack 2 Stack: Empty
Stack 3 Stack: Empty
Stack 4 Stack: Empty
Power Stack: Empty
Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': 2
Sort the 'B' from the list.
List of Elements to Sort: (B) B G Y B G Y B Y B R R Y
Stack Options:
Stack 1 Stack: Y
Stack 2 Stack: G
Stack 3 Stack: Empty
Stack 4 Stack: Empty
Power Stack: Empty
Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': 3
Sort the 'B' from the list.
List of Elements to Sort: (B) G Y B G Y B Y B R R Y
Stack Options:
Stack 1 Stack: Y


Microsoft Visual Studio Debug Console
List of Elements to Sort: (R) R Y
Stack Options:
Stack 1 Stack: Y Y Y Y
Stack 2 Stack: G G G
Stack 3 Stack: B B B B B
Stack 4 Stack: Empty
Power Stack: Empty
Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': 4
Sort the 'R' from the list.
List of Elements to Sort: (R) Y
Stack Options:
Stack 1 Stack: Y Y Y Y
Stack 2 Stack: G G G
Stack 3 Stack: B B B B B
Stack 4 Stack: R
Power Stack: Empty
Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': 4
Sort the 'Y' from the list.
List of Elements to Sort: (Y)
Stack Options:
Stack 1 Stack: Y Y Y Y
Stack 2 Stack: G G G
Stack 3 Stack: B B B B B
Stack 4 Stack: R R
Power Stack: Empty
Enter stack choice (1/2/3/4) or '5' to quit and to use special power enter '6': 1
All Sorted!
| YOUR FINAL SCORE: 75 |
Would you like to continue with the game? If not enter 0:0
Thank you for playing :D
C:\Users\Humal Hassan\source\repos\Project3\x64\Debug\Project3.exe (process 8756) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

FUTURE WORK:

For future enhancements, the Color Sorting Game could undergo several improvements to enrich the gaming experience. Potential developments include refining the user interface with graphical enhancements, introducing a multiplayer mode for interactive gameplay, incorporating additional difficulty levels, and allowing users to customize game settings. Implementing features such as leaderboards, educational components related to color theory, and mobile compatibility could further enhance the game's appeal and accessibility. Additionally, refining randomization algorithms, adding persistence for game progress, and addressing accessibility considerations would contribute to a more polished and inclusive gaming experience. Bug fixing, performance optimization, and potential localization efforts could ensure a smoother and more enjoyable experience for players.

CONCLUSION:

the Color Sorting Game is a fun and challenging project that allows users to enhance their color sorting skills. Built in C++, the game features different difficulty levels, engaging gameplay, and a user-friendly interface. Through strategic sorting and decision-making, players can enjoy an entertaining and educational experience. With potential future updates, the game has the flexibility to evolve and offer continued enjoyment for a diverse audience.

REFERENCE WORK:

https://stackoverflow.com/questions/50047872/color-game-calling-random-from-array-c