STEP 1: Problem Identification and Statement

Problem Statement:
Play the Rock Paper Scissors Game exactly five times against a computer that randomly generates outcomes

STEP 2: Gathering Information and Input/Output diagram

General Description of Rock Paper Scissors Game (RPS):
RPS is a simple hand game typically played between two people. The game has three possible outcomes: Rock crushes Scissors, Scissors cuts Paper, and Paper covers Rock. Each of the three basic moves defeats one of the other two, and it is a tie if both players choose the same move.

We want to play the RPS Game five times against a computer that randomly generates RPS at each time. In order to do so, there is some necessary information that we need to identify first. Those are listed below as:

- User's selection of Rock, Paper, Scissors (RPS) for five times
  We asks users to input their selection of RPS, and store each selection into the array

- Computer's randomly generated selection of RPS for five times
  We store each selection into the array

Figure 1: Examples of Rock, Paper, and Scissors

- Rules for the RPS game (In the RPS game, there are certain rules for how to win, lose, and tie.)
  Win
  - Rock wins against Scissors
  - Scissors win against Paper
  - Paper wins against Rock

  Lose
  - Rock loses to Paper
  - Scissors loses to Rock
  - Paper loses to Scissors

Tie

- If both selections are the same, the result is a tie. (Eg. rock and rock result in a tie)

The diagram below shows an input/output (I/O) diagram, where the black box represents the computer program.

The inputs are : five rounds of user's selections and computer's randomly generated selections
The outputs are : a summary of the game with who wins and how many times an user wins, loses, and ties

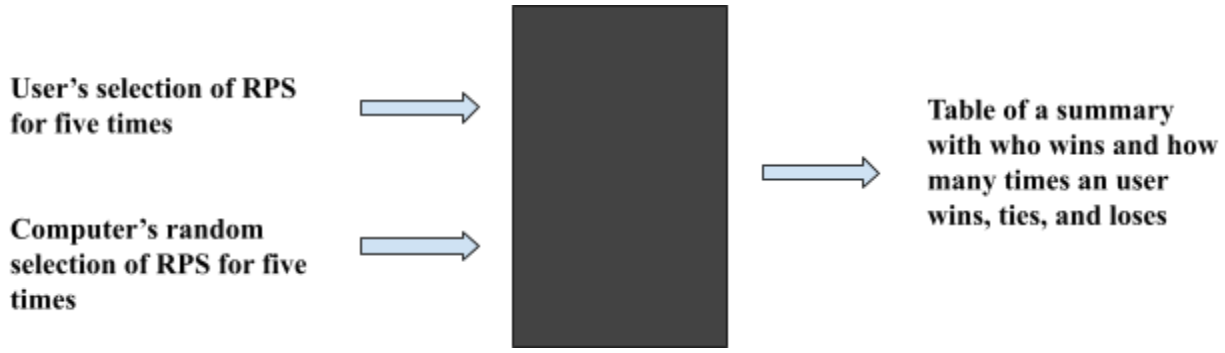

Figure 2: I/O Diagram

Menu description for I/O diagram

1. A user is asked to select among three choices: 1 for Rock, 2 for Paper, or 3 for Scissors, five times consecutively. → Input
2. At the same time, a computer will randomly generate its own selections among those three choices. The result will be shown each round. → Input
3. After five rounds, the table of results with who wins and how many times a user wins, loses, and ties will be displayed at the end. → Output

STEP 3: Test Cases and Algorithm Design

In order to play an RPS game five times and obtain a summary of the game that shows who wins and how many times an user wins, loses, and ties, there are certain steps that we have to follow.

1. Choose selections

User's selections of RPS

- We ask them to choose 1 for rock, 2 for paper, or 3 for scissors five times in total.

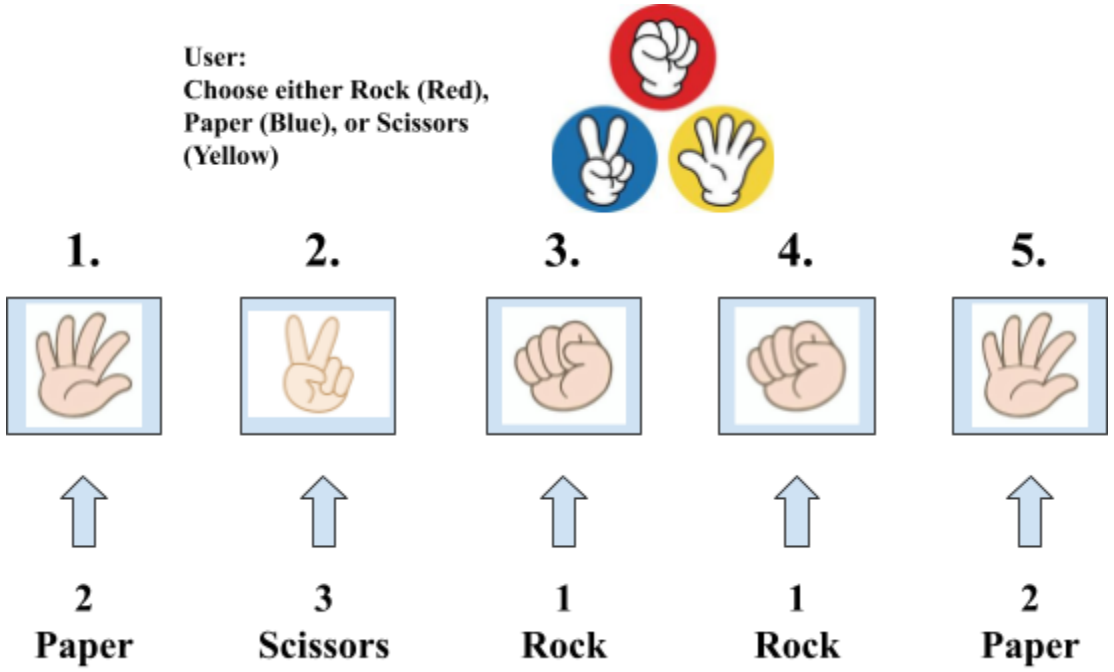

Figure 3: Example of User's Inputs for five times

Computer's selections of RPS

- Their selections will be generated randomly by using the seed of the Random Number Generator and the C++ rand () function
- The computer will randomly choose 1 for rock, 2 for paper, or 3 for scissors

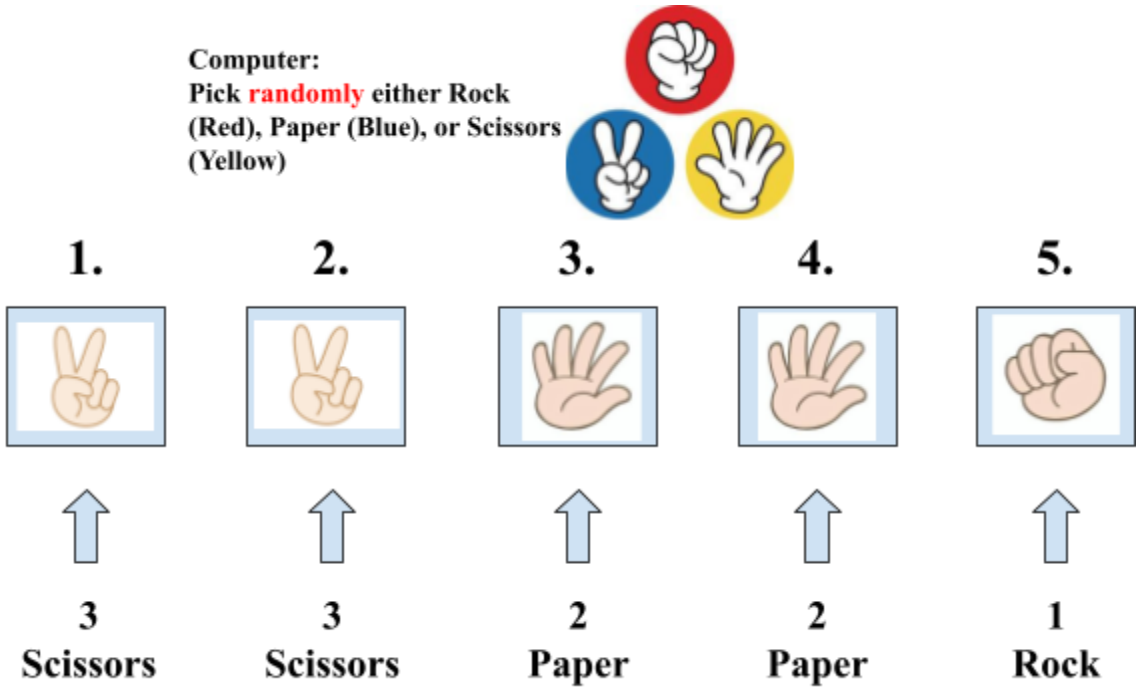

Figure 4: Example of Random Selections by Computer for five times

2. Play RPS game based upon the Rules for the RPS game

In the two figures above, in the first round, the user chose paper and the computer picked scissors. In this case, following the rules, the user will lose to the computer. In the second round, however, both the user and computer chose scissors, meaning that the game is a tie. Furthermore, in the fifth round, the user selected paper and the computer chose rock, which means that the user will win against the computer based on the rules.

3. Table of Results

After five consecutive rounds, we want to obtain the table of a summary. The expected table of a summary is illustrated below as:

You: paper	**Computer: scissors**	**Lose**
You: scissors	**Computer: scissors**	**Tie**
You: rock	**Computer: paper**	**Lose**
You: rock	**Computer: paper**	**Lose**
You: paper	**Computer: rock**	**Win**
	You lose (win:1, tie:1, lose:3)	

Figure 5: Example of the Table of Results after five rounds

Final Results

According to Figure 5, since the number of losses outweighs the number of wins, the final result is that the user lost. (If the number of wins or losses is the same, then the result will be a tie for the user.)

Test Cases

The following provides a set of test cases that can be used to test the algorithm and software. It is advisable to verify the program by testing each case, including wins, losses, and ties. Additionally, using values smaller than 1 and larger than 3 to check if the program fails or not is recommended.

Case 1: Win

We would like to check if a user wins when the number of wins outweighs that of loses, which is the case when **Wins > Loses**

For the test case,

Wins = 3 and Loses = 1 → the expected outcome will be win.

Case 2: Lose

We would like to check if a user loses when the number of loses outweighs that of wins, which is the case when **Loses > Wins**

For the test case,
Wins = 2 and Loses = 3 → the expected outcome will be lose.

Case 3: Tie
We would like to check if a user ties when the number of wins is equal to that of loses, which is the case when **Wins = Loses**
For the test case,
Wins = 2 and Loses = 2 → the expected outcome will be tie.

Case 4: Invalid Input smaller than 1
Input = 0 → the expected outcome will be invalid.

Case 5: Invalid input larger than 3
Input = 4 → the expected outcome will be invalid.

Case 6: Termination
Input = 3 → the expected outcome will be that the program will terminate.

Algorithm

The algorithm can be expressed as:
Declare function prototypes for print_Menu as integer, play_RPS as void, print_Summary as void,

Define the main function
    Declare size = 5 as a constant integer,
    Declare arrays, player of size "size" and computer of size "size", as integers,

    Initialize the seed of the Random Number Generator,

    Declare choice as an integer,
    Repeat
        Call the function print_Menu and set it equal to choice,

        If choice is equal to 1,
            Call the void function play_RPS,
            Call the void function print_Summary,

        Otherwise if choice is equal to 2,
            Print "Help: Rock beats Scissors, Scissors beats Paper, Paper beats Rock."

        Otherwise if choice is equal to 3,
            Print "Terminating …"

        Otherwise
            Print "Invalid input. Please enter 1, 2, or 3: "

While logical expression, which is choice is not equal to 3, is true

Return to 0

Define the function print_Menu
Declare choice as an integer,
Print “Welcome to the Rock Paper Scissors game. You will play 5 times…”
Print “1) Play”
Print “2) Help”
Print “3) Exit”
Print “Please make a selection (1-3)”
Read value into choice
Repeat while logical expression, which is choice is smaller than 1 or choice is larger than 3, is true
Print “Invalid input. Please enter 1, 2, or 3: “
Read value into choice

Return to choice

Define the void function, play_RPS, with arrays, player and computer, and size as integer
Print “Please select among 1) Rock, 2) Paper, or 3) Scissors: “
Repeat for each round in the range from 1 to size (initialize integer i as 1 and i will increment by 1 each time until one before size)
Read value into player [i]
Repeat while logical expression, which is player [i] is smaller than 1 or player [i] is larger than 3, is true
Print “Invalid input. Please enter 1, 2, or 3: “
Read value into player[i]

Assign random number between 1 and 3 to computer [i] by setting it equal to C++ rand() function
Print “You: “
If player [i] is equal to 1,
`Print “Rock”

Otherwise if player [i] is equal to 2,
Print “Paper”

Otherwise,
Print “Scissors”

Print “Computer: “
If computer [i] is equal to 1,

`Print “Rock”

Otherwise if computer [i] is equal to 2,
Print “Paper”

Otherwise,
Print “Scissors”

Print a last line

Define the void function, print_Summary, with arrays, player and computer, and size as integer
Declare wins, loses, and ties as integers, and initialize them to 0
Print “Summary:”
Repeat for each round in the range from 1 to size (initialize integer i as 1 and i will increment by 1 each time until one before size)
Print “You: “
If player [i] is equal to 1,
`Print “Rock”

Otherwise if player [i] is equal to 2,
Print “Paper”

Otherwise,
Print “Scissors”

Print “Computer: “
If computer [i] is equal to 1,
`Print “Rock”

Otherwise if computer [i] is equal to 2,
Print “Paper”

Otherwise,
Print “Scissors”

If player [i] is equal to 1 and player [i] is equal to 2,
Print “ - Lose “
Post-Increment loses by 1

Otherwise if player [i] is equal to 2 and player [i] is equal to 3,
Print “ - Lose “
Post-Increment loses by 1

Otherwise if player [i] is equal to 3 and player [i] is equal to 1,

Print " - Lose "
Post-Increment loses by 1

Otherwise if player [i] is equal to computer [i],
Print " - Tie "
Post-Increment ties by 1

Otherwise
Print " - Win "
Post-Increment wins by 1

Print a last line

If wins is larger than loses,
Print "You win!", " (win: ", wins, " tie: ", ties, " lose: ", loses, ")"

Otherwise if loses is larger than wins,
Print "You lose", " (win: ", wins, " tie: ", ties, " lose: ", loses, ")"

Otherwise
Print "You tie", " (win: ", wins, " tie: ", ties, " lose: ", loses, ")"

STEP 4: Implementation

```
/*-------------------------------------------------------------------------------------
-------*/
/* Name: Shota Matsumoto, Student Number: sm11745*/
/* Date: March 20, 2024*/
/* Program: CPE Second Assignment - Shota Matsumoto.cpp*/
/* Description: This program executes a Rock Paper Scissors Game 5 times between a
user and computer.*/
/*
-------------------------------------------------------------------------------------
----*/
#include <iostream>
#include <ctime>
using namespace std;
// declare function prototypes
int print_Menu();
void play_RPS(int player[], int computer[], int size);
void print_Summary(int player[], int computer[], int size);
//Define the main fuction
int main(){
    //Declare variables and initialize size to 5
    const int size = 5;
    //Declare the arrays
    int player[size], computer[size];

    //Sets the seed of the Random Number Generator
```

```
    srand(time(NULL));

    int choice;
    //Insert do/while Repetition structure to evaluate the condition at the end
    do {
        //Call the function, print_Menu, to get a user's selection
        choice = print_Menu();

        //Make if statement. Here on the top we deal with the case of choice 1.
        if (choice == 1){
            play_RPS(player, computer, size); //Call the function, play_RPS, to play
the RPS game
            print_Summary(player, computer, size); //Call the function, print_Summary,
to print a summary of the game
        }

        //Make else/if statement for the choice being 2. Print the information about
the program and re-display main menu.
        else if (choice == 2){
            cout << "Help: Rock beats Scissors, Scissors beats Paper, Paper beats
Rock." << endl;
        }

        //Make else/if statement for the choice being 3. This terminates the program.
        else if (choice == 3){
            cout << "Terminating ..." << endl;
        }

        //Make else statement for values other than 1, 2, or 3. Allow users to input
the value again from main menu.
        else {
            cout << "Invalid input. Please enter 1, 2, or 3: " << endl;
        }
    } while (choice != 3); //Provide the condition for do/while Repepition structure

    return 0;
}
//Define the function
int print_Menu() {
    int choice;
    //Print Main Menu and three choices and Read choice
    cout << "Welcome to the Rock Paper Scissors game. You will play 5 times…" << endl;
    cout << "1) Play" << endl;
    cout << "2) Help" << endl;
    cout << "3) Exit" << endl;
    cout << "Please make a selection (1-3): ";
    cin >> choice;
    //Insert While Repitition structure to allow users to input a value again.
    while (choice < 1 || choice > 3) {
        cout << "Invalid choice. Please enter 1, 2, or 3: ";
        cin >> choice;
    }
    return choice; //Make sure to return to choice
}
//Define the void function
```

```
void play_RPS(int player[], int computer [], int size){
    cout << "Please select among 1) Rock, 2) Paper, or 3) Scissors: " << endl;
    //Insert for looping structure. Declare and Initialize i which is a counter.
Provide conditions to the counter, and post-increment 1 every round
    for (int i=0; i < size; i++){
        cin >> player[i];
        //Insert While Repitition structure to allow users to input a value again.
        while (player[i] < 1 || player[i] > 3){
            cout << "Invalid input. Please enter 1, 2, or 3: " << endl;
            cin >> player[i];
        }
        //randomly generate the integer between 1 and 3
        computer[i] = rand () % 3 + 1;
        cout << "You: ";

        //Make if statements to print Rock for 1, Paper for 2, or Scissors for 3
        if (player[i] == 1){
            cout << "Rock";
        }
        else if (player[i] == 2){
            cout << "Paper";
        }
        else {
            cout << "Scissors";
        }
        cout << " Computer: ";
        if (computer[i] == 1){
            cout << "Rock";
        }
        else if (computer[i] == 2){
            cout << "Paper";
        }
        else {
            cout << "Scissors";
        }
        cout << endl;
    }
}
//Define the void function
void print_Summary(int player[], int computer[], int size){
    //Declare variables and Initialize them to 0
    int wins = 0, loses = 0, ties = 0;
    cout << "Summary: " << endl;
    //Insert for looping structure. Declare and Initialize i which is a counter.
Provide conditions to the counter, and post-increment 1 every round
    for (int i = 0; i < size; i++){
        cout << "You: ";
        //Make if statements to print Rock for 1, Paper for 2, or Scissors for 3
        if (player[i] == 1){
            cout << "Rock";
        }
        else if (player[i] == 2){
            cout << "Paper";
        }
        else {
```

```
            cout << "Scissors";
        }
        cout << " Computer: ";
        if (computer[i] == 1){
            cout << "Rock";
        }
        else if (computer[i] == 2){
            cout << "Paper";
        }
        else {
            cout << "Scissors";
        }
        //Make if statements to print Win, Lose, or Tie based on the power relations
in the RPS game and Post-increment a variable (wins, loses, or ties) by 1 if selected
        if (player[i] == 1 && computer [i] == 2){
            cout << " - Lose";
            loses++;
        }
        else if (player[i] == 2 && computer [i] == 3){
            cout << " - Lose";
            loses++;
        }
        else if (player[i] == 3 && computer [i] == 1){
            cout << " - Lose";
            loses++;
        }
        else if (player[i] == computer [i]){
            cout << " - Tie";
            ties++;
        }
        else {
            cout << " - Win";
            wins++;
        }
        cout << endl;
    }

    //Make if statements to print who wins and how many times an user wins, loses, and
ties
    if (wins > loses){
        cout << "You win!" << " (win: " << wins << " , tie: " << ties << " , lose: "
<< loses << ")" << endl;
    }

    else if (wins < loses){
        cout << "You lose" << " (win: " << wins << " , tie: " << ties << " , lose: "
<< loses << ")" << endl;
    }

    else {
        cout << "You tie" << " (win: " << wins << " , tie: " << ties << " , lose: " <<
loses << ")" << endl;
    }
}
```

STEP 5: Tests and Verification

Test Case 1:
The number of wins is 3 and the number of loses is 1.
The output is win.
This is in agreement with the expected outcome, which is win. This proves the condition where the outcome must be win when wins > loses.

```
Summary:
You: Paper Computer: Paper - Tie
You: Scissors Computer: Paper - Win
You: Paper Computer: Rock - Win
You: Rock Computer: Scissors - Win
You: Paper Computer: Scissors - Lose
You win! (win: 3 , tie: 1 , lose: 1)
Welcome to the Rock Paper Scissors game. You will play 5 times…
1) Play
```

Figure 6: The output for test case 1

Test Case 2:
The number of wins is 2 and the number of loses is 3.
The output is lose.
This is in agreement with the expected outcome, which is lose. This proves the condition where the outcome must be lose when loses > wins.

```
Summary:
You: Rock Computer: Scissors - Win
You: Paper Computer: Rock - Win
You: Scissors Computer: Rock - Lose
You: Paper Computer: Scissors - Lose
You: Scissors Computer: Rock - Lose
You lose (win: 2 , tie: 0 , lose: 3)
Welcome to the Rock Paper Scissors game. You will play 5 times…
1) Play
```

Figure 7: The output for test case 2

Test Case 3:
The number of wins and loses is equal to each other, being 2 and 2.
The output is tie.
This is in agreement with the expected outcome, which is tie. This proves the condition where the outcome must be tie when wins = loses.

```
Summary:
You: Paper Computer: Paper - Tie
You: Scissors Computer: Rock - Lose
You: Paper Computer: Scissors - Lose
You: Rock Computer: Scissors - Win
You: Rock Computer: Scissors - Win
You tie (win: 2 , tie: 1 , lose: 2)
Welcome to the Rock Paper Scissors game. You will play 5 times…
1) Play
```

Figure 8: The output for test case 3

Test Case 4:
Input value is 0.
The output is invalid.
This is in agreement with the expected outcome, which is invalid.

```
Welcome to the Rock Paper Scissors game. You will play 5 times…
1) Play
2) Help
3) Exit
Please make a selection (1-3): 1
Please select among 1) Rock, 2) Paper, or 3) Scissors:
0
Invalid input. Please enter 1, 2, or 3:
```

Figure 9: The output for test case 4

Test Case 5:
Input value is 4.
The output is invalid.
This is in agreement with the expected outcome, which is invalid.

```
Welcome to the Rock Paper Scissors game. You will play 5 times
1) Play
2) Help
3) Exit
Please make a selection (1-3): 1
Please select among 1) Rock, 2) Paper, or 3) Scissors:
4
Invalid input. Please enter 1, 2, or 3:
```

Figure 10: The output for test case 5

Test Case 6:
Input value is 3.
The output is that the program terminated.
This is in agreement with the expected outcome, which is that the program will terminate.

```
1) Play
2) Help
3) Exit
Please make a selection (1-3): 3
Terminating ...


...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 11: The output for test case 6

Conclusion

Based upon the outcomes of these 6 various test cases, we can conclude that the program functions successfully and allows users to play the RPS game five times against a computer properly.

User's Guide

- To execute the program, compile and run the code found in the file named CPE First Assignment.cpp

1. Welcome Message and Main Menu:
    - When you run the program, it will display a welcome message along with the main menu.
    - The main menu has three options:
        - Play (1): Select this option to start playing the Rock Paper Scissors game against the computer.
        - Help (2): Select this option to view information about the rules of the game.
        - Exit (3): Select this option to terminate the program.

2. Playing the Game (Option 1):
    - After choosing to play (option 1), you will play the game five times consecutively against the computer.
    - For each round, you will be prompted to choose Rock (1), Paper (2), or Scissors (3).
    - Enter your choice by typing the corresponding number and pressing Enter.
    - The computer will randomly generate its choice for each round.
    - After each round, the program will display your choice and the computer's choice, indicating who won the round (Win), lost the round (Lose), or tied (Tie).

3. Game Summary (Option 1):
    - After playing five rounds, the program will display a summary of the game.
    - The summary will show the choices made by both you and the computer for each round, along with the outcome (Win, Lose, or Tie) of each round.
    - Additionally, the summary will show the total number of wins, losses, and ties throughout the five rounds.
    - Finally, the program will declare whether you won, lost, or tied the overall game based on the number of wins, losses, and ties accumulated.

4. Help Information (Option 2):
    - If you choose the Help option (2) from the main menu, the program will display information about the rules of the Rock Paper Scissors game.
    - It will remind you that Rock beats Scissors, Scissors beats Paper, and Paper beats Rock.

5. Exiting the Program (Option 3):
    - If you choose to exit (option 3) from the main menu, the program will terminate.

6. Invalid Input Handling:
    - The program handles invalid inputs gracefully. If you enter a value outside the valid range or a non-numeric input, it will prompt you to enter a valid choice again.

7. Repetition and Looping:
    - The program uses loops (do-while, for) to repeat certain actions such as displaying the main menu, playing multiple rounds, and handling user input until a valid choice is made or the program is terminated.rom the main menu terminates the program.