****

**INFORMATICS INSTITUTE OF TECHNOLOGY**

In Collaboration with

**UNIVERSITY OF WESTMINSTER (UOW)**

**BEng (Hons) Software Engineering**

**Computer Science Practice**

**4COSC008C**

**Student ID: 2016323**

**Student Name: C.Kajendran**

Table of Contents

Contends Page no

[**Acknowledgement** 2](#_Toc478299477)

[**Introduction 3**](#_Toc478299478)

[**Analysis** 4](#_Toc478299479)

[**Functions** 5](#_Toc478299480)

[**Code** 11](#_Toc478299481)

[**Testing** 20](#_Toc478299482)

[**Conclusion 22**](#_Toc478299483)

## Acknowledgement

I would like to thank Mr. Guganathan for the guidance and immense support given towards this Programming Methodology module and tolerating all the time.

Finally I am really thankful to my parents and my friends for being patient and encouraging me continuously.

# Introduction

* In this Temple run course work twenty players are compete for three winning titles.
* The winners are selected based on distance they run, number of coins that they have collected, scores that they have achieved in the game.
  + - Winner 1 – Competitor who reached maximum distance.
    - Winner 2 – Competitor who collected maximum number of coins.
    - Winner 3 – Competitor who achieved highest score.

By considering with twenty players, overall winners are selected to the above three titles.

* This program need to run till the user wishes to quit.
* This program should run contains to view selected player details and selected winners type personally too.

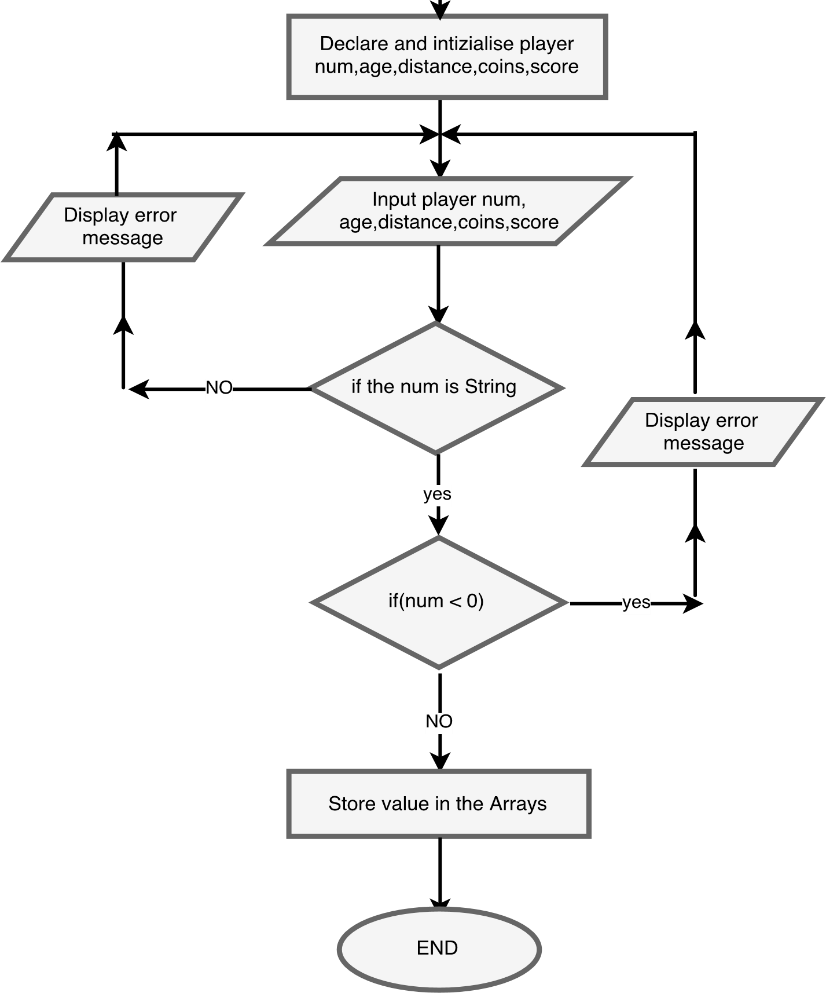
# Analysis

**Functional requirements**

* First user should get twenty competitors details includes ,
  + - Competitor’s number.
    - Competitor’s name.
    - Competitor’s age.
    - Competitor’s distance.
    - Competitor’s coins.
    - Competitor’s score.
* All the above mentioned details are stored in arrays, using user defined data types and need to be validated.
* After storing the inputs three types of winners are need to calculate and display,
  + Competitor with maximum distance.
  + Competitor with maximum coin collection.
  + Competitor with highest score.
* Using menu option to view ,
  + View competitor details.
  + View winners of all three types.
  + View the selected type of winner (with in winner in distance, coins and score).
  + View the selected competitor’s details.
* Program ne to continue after every using these all option and it need to end when the user is wishing to end.

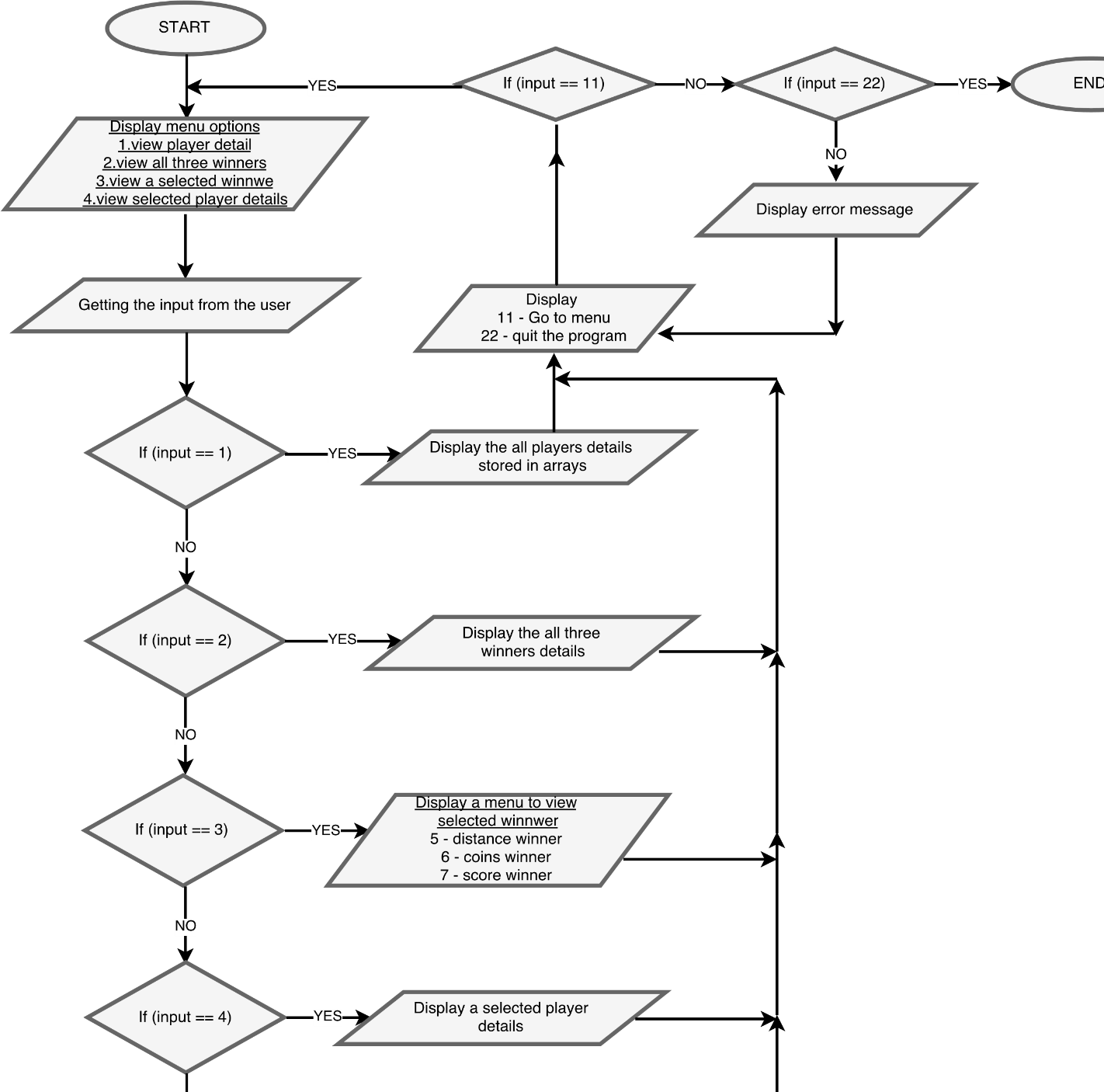
# Functions

1. Getting the input from the user and validating and then storing the details.

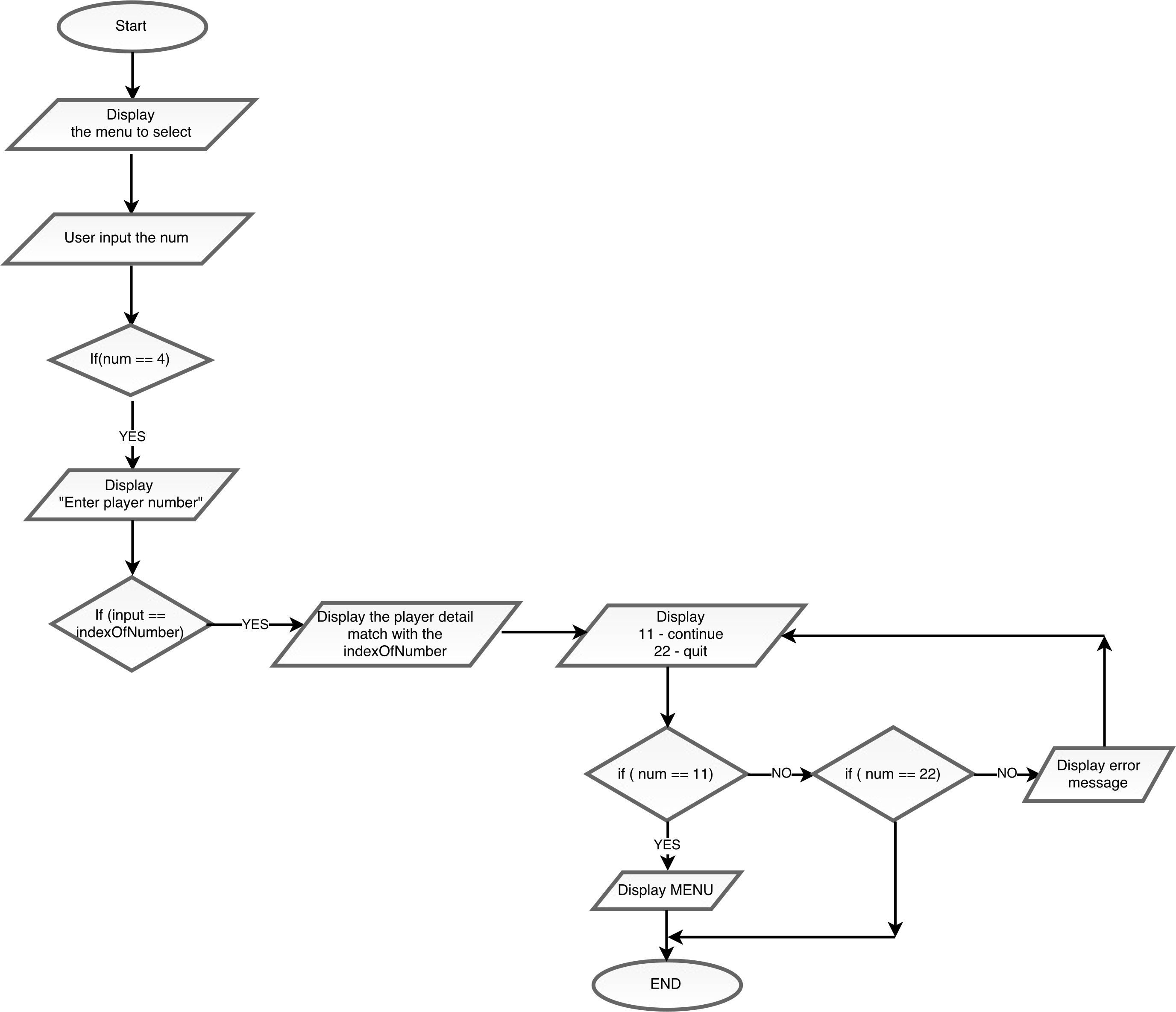


For all the other inputs such as player age, distance, coins, score are the same way and same validation.

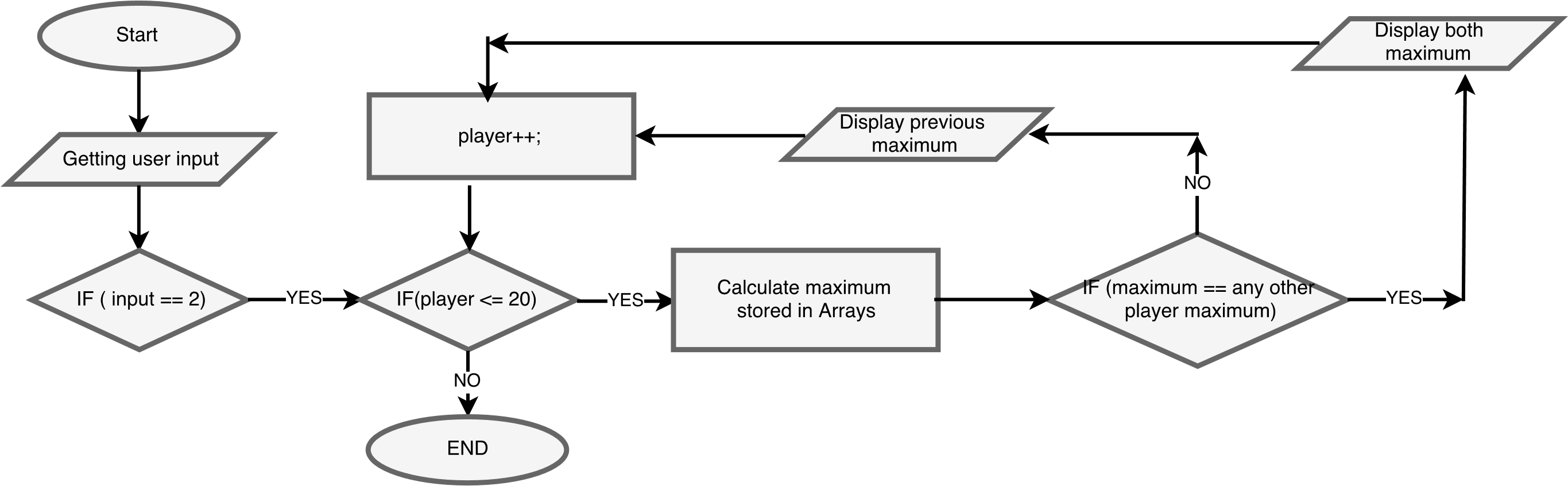
1. Displaying the menu and displaying the winner’s details



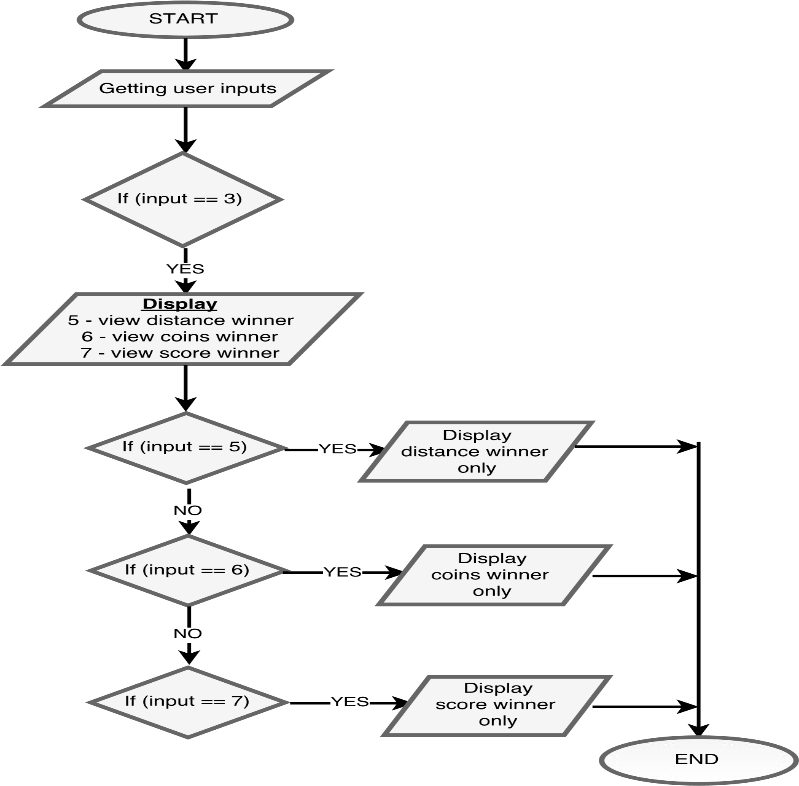
1. Details of selected player details



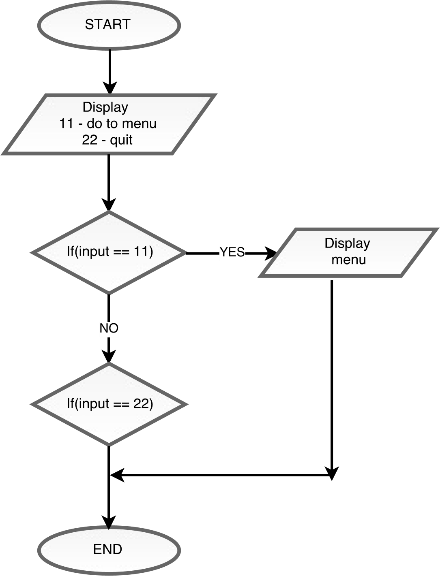
1. Finding the winner of distance, coins and score



1. Viewing the selected category of winner



1. Ending and returning to menu



# Code

1. Getting the input ,validating and storing in Arrays

public void userInput() {

System.out.println(" > Fill the above questions to enter player details < " + "\n");

for (int player = 0; player < 3; player++) {

System.out.println((player + 1) + "]");

do {

System.out.println(" [01] Enter player number : "); //player number

if (input.hasNextInt()) {

playerNum = input.nextInt();

IsNumber = true;

if (playerNum < 0) {

isNumber = false;

System.err.println(" >ERROR Enter a positive number<" + "\n");

System.out.println(" [01] Enter player number : ");

playerNum = input.nextInt();

}

if (playerNum > 0) {

isNumber = true;

}

} else {

isNumber = false;

Input.next();

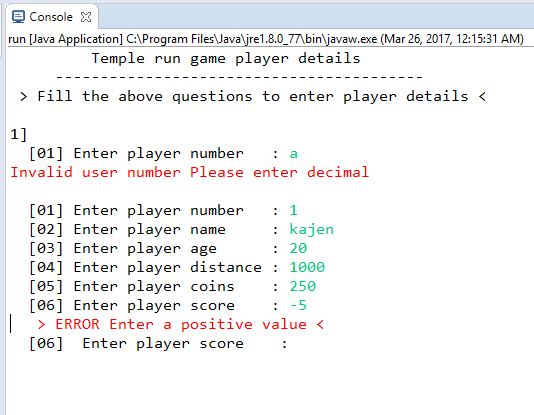
System.err.println("Invalid user number Please enter decimal" + "\n");

}

} while (!(isNumber));

}

}



For all the other inputs such as player age, distance, coins, score are the same way and same validation

1. Displaying the menu details

public void display() {

System.out.println(" Select the above options to view details");

System.out.println("------------------------------------------" + "\n");

System.out.println("[1] - Veiw player details" + "\n" + "[2] - view winner details" + "\n" +

[3] - view selected winners" +"\n" + " [4] - View selected player details");

user = input.next();

playerDetails();

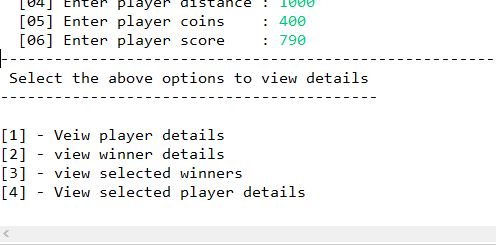
winnerDetails();

selectWinner();

selectedPlayer();

goToMenu();

}



1. Displaying all players’ details.

public void playerDetails() {

switch (i) {

case "1":

System.out.println(" Player Details");

for (int user = 0; user < 3; user++) {

System.out.println("Player number is : " + " " + collectionOfPlayers[user].playerNum);

System.out.println("Player name is : " + " " + collectionOfPlayers[user].playerName);

System.out.println("Player age is : " + " " + collectionOfPlayers[user].playerAge);

System.out.println("Player distance is : " + " " + collectionOfPlayers[user].playerDistance);

System.out.println("Player coin collection is : " + " " + collectionOfPlayers[user].playerCoins);

System.out.println("Player score is : " + " " + collectionOfPlayers[user].playerScore);

System.out.println("--------------------------------------------------------------");

}

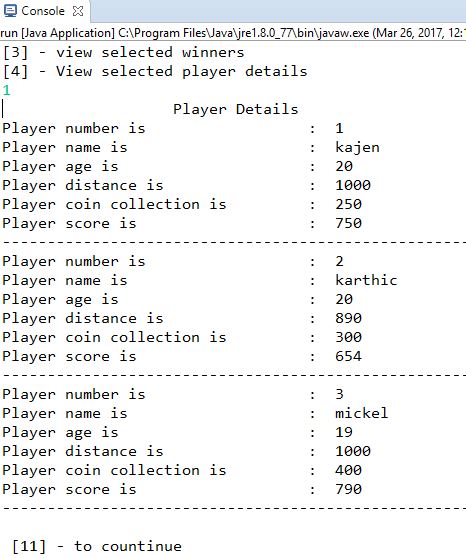
System.out.println("\n" + " [11] - to countinue" + "\n" + "[22] - END");

user = input.next();

break;

}

}



1. Displaying all three winners’ details

public void winnerDetails() {

switch (i) {

case "2":

System.out.println("Winner of three type methods");

System.out.println("---------------------------------------" + "\n");

int largest = collectionOfPlayers[0].playerDistance; //calculation to find the highest distances

for (int num = 0; num < collectionOfPlayers.length; num++) {

if (playerDistance > largest) {

largest = collectionOfPlayers[num].playerDistance;

}

}

for (int sameValue = 0; sameValue < 3; sameValue++) {

if (largest == collectionOfPlayers[sameValue].playerDistance) {

System.out.println(" (01] Name of the player: " + collectionOfPlayers[sameValue].playerName);

System.out.println(" (02] Age of the player is: " + collectionOfPlayers[sameValue].playerAge);

System.out.println(" (03] Player distance is : " + largest);

System.out.println("---------------------------------------------------");

}

}

int max = collectionOfPlayers[0].playerCoins;

for (int num = 0; num < collectionOfPlayers.length; num++) {

if (playerCoins > max) {

max = collectionOfPlayers[num].playerCoins;

}

}

for (int samecoin = 0; samecoin < 3; samecoin++) {

if (max == collectionOfPlayers[samecoin].playerCoins) {

System.out.println(" (01] Name of the player: " + collectionOfPlayers[samecoin].playerName);

System.out.println(" (02] Age of the player : " + collectionOfPlayers[samecoin].playerAge);

System.out.println(" (03] Player coin cllection is : " + max);

System.out.println("---------------------------------------------------");

}

}

int highest = collectionOfPlayers[0].playerScore;

for (int num = 0; num < collectionOfPlayers.length; num++) {

if (playerScore > highest) {

highest = collectionOfPlayers[num].playerScore;

}

}

for (int sameValue = 0; sameValue < 3; sameValue++) {

if (highest == collectionOfPlayers[sameValue].playerScore) {

System.out.println(" (01] Name of the player: " + collectionOfPlayers[sameValue].playerName);

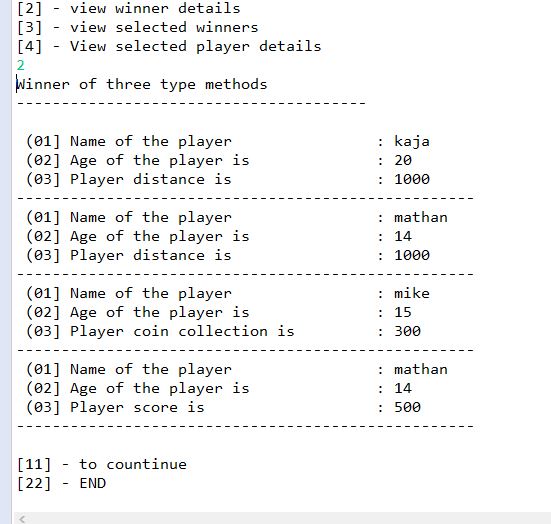
System.out.println(" (02] Age of the player : " + collectionOfPlayers[sameValue].playerAge);

System.out.println(" (03] Player score is : " + highest);

System.out.println("---------------------------------------------------");

}

}



1. Displaying selected winner details

private void selectWinner() {

switch (i) {

case "3":

System.out.println("\n [6] - Winner of distance" + "\n [7] - Winner of coin collection" + "\n [8] -Winner of highest score");

String in = input.next();

switch ( in ) {

case "6":

int largest = collectionOfPlayers[0].playerDistance; //calculation to find the highest distances

for (int num = 0; num < collectionOfPlayers.length; num++) {

if (playerDistance > largest) {

largest = collectionOfPlayers[num].playerDistance;

}

}

for (int sameValue = 0; sameValue < 3; sameValue++) {

if (largest == collectionOfPlayers[sameValue].playerDistance) {

System.out.println(" (01] Name of the player: " + collectionOfPlayers[sameValue].playerName);

System.out.println(" (02] Age of the player : " + collectionOfPlayers[sameValue].playerAge);

System.out.println(" (03] Player distance is : " + largest);

System.out.println("---------------------------------------------------");

}

}

System.out.println("\n" + "[11] - to countinue" + "\n" + "[22] - END");

i = input.next();

break;

case "7":

int max = collectionOfPlayers[0].playerCoins;

for (int num = 0; num < collectionOfPlayers.length; num++) {

if (playerCoins > max) {

max = collectionOfPlayers[num].playerCoins;

}

}

for (int sameValue = 0; sameValue < 3; sameValue++) {

if (max == collectionOfPlayers[sameValue].playerCoins) {

System.out.println(" (01] Name of the player : " + collectionOfPlayers[sameValue].playerName);

System.out.println(" (02] Age of the player : " + collectionOfPlayers[sameValue].playerAge);

System.out.println(" (03] Player distance is : " + max);

System.out.println("---------------------------------------------------");

}

}

System.out.println("\n" + "[11] - to countinue" + "\n" + "[22] - END");

user = input.next();

break;

case "8":

int highest = collectionOfPlayers[0].playerScore;

for (int num = 0; num < collectionOfPlayers.length; num++) {

if (playerScore > highest) {

highest = collectionOfPlayers[num].playerScore;

}

}

for (int sameValue = 0; sameValue < 3; sameValue++) {

if (highest == collectionOfPlayers[sameValue].playerScore) {

System.out.println(" (01] Name of the player: " + collectionOfPlayers[sameValue].playerName);

System.out.println(" (02] Age of the player : " + collectionOfPlayers[sameValue].playerAge);

System.out.println(" (03] Player distance is : " + highest);

System.out.println("---------------------------------------------------");

}

}

System.out.println("\n" + "[11] - to countinue" + "\n" + "[22] - END");

i = input.next();

break;

default:

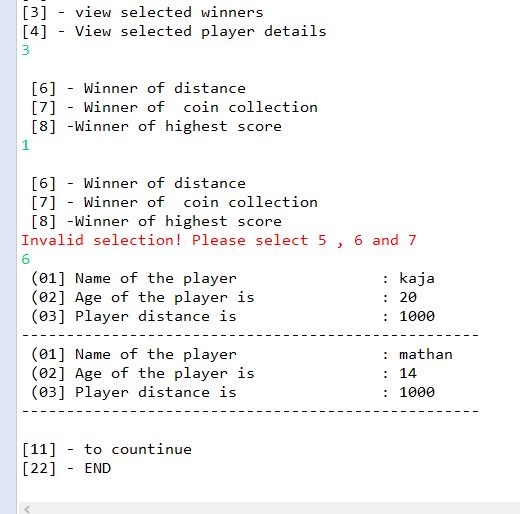
System.err.println("Invalid selection! Please select 5 , 6 and 7");

selectWinner();

i = input.next();

}

}

}

1. Display selected player details

public void selectedPlayer() {

switch (i) {

case "4":

System.out.println("\n Enter player number:");

int detail = input.nextInt();

System.out.println("Player number is : " + collectionOfPlayers[detail - 1].playerNum);

System.out.println("Player name is : " + collectionOfPlayers[detail - 1].playerName);

System.out.println("Player age is : " + collectionOfPlayers[detail - 1].playerAge);

System.out.println("Player distance : " + collectionOfPlayers[detail - 1].playerDistance);

System.out.println("Player coin collection : " + collectionOfPlayers[detail - 1].playerCoins);

System.out.println("Player score is : " + collectionOfPlayers[detail - 1].playerScore);

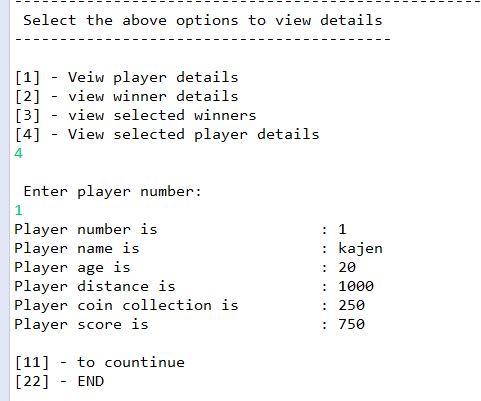
System.out.println("\n" + "[11] - to countinue" + "\n" + "[22] - END");

user = input.next();

break;

}

}



1. Ending program and returning to the menu

**public** **void** goToMenu(){

**switch**(i){

**case** "11":

display();

**break**;

**case** "22":

input.close();

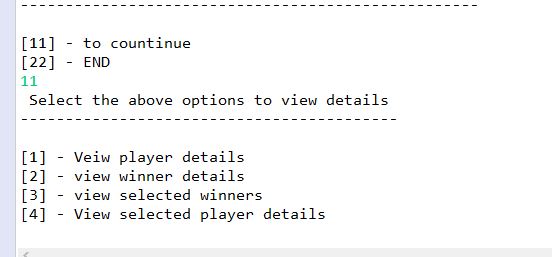
**break**;

**default**: System.***err***.println("Invalid selection" + "\n");

}

}

}



## Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test no. | Input data | Expected output | Actual output | | Status  (ok / bug) |
| 1 | Input player number : 1 | Save and display next question | Save and display next question | | ok |
| 2 | Input player age : 20 | Save and display next question | Save and display next question | | ok |
| 3 | Input player distance : 1000 | Save and display next question | Save and display next question | | ok |
| 4 | Input player coins : 100 | Save and display next question | Save and display next question | | ok |
| 5 | Input player score : 41 | Save and display next question | Save and display next question | | Ok |
| 6 | menu : 1 | Display option 1 | Display option 1 | | Ok |
| 7 | Menu : 2 | Display option 2 | Display option 1 | | Ok |
| 8 | Selected winner : 6 | Display distance winner | Display distance winner | | ok |
| 9 | Returning to menu : 11 | Display menu | Display menu | | ok |
| 10 | Quit program : 22 | End program | Error and crashing | | bug |
| 11 | Input player number : a | Display error message | Display error message | | ok |
| 12 | Input player number : -20 | Display error message | Display error message | | ok |
| 13 | Menu : 1 | Display details | Display details | | ok |
| 14 | Selected winner : 3 | Display Winners’ type to select | Display Winners type to select | | ok |
| 15 | Selected player : p | Display error message | Display error message | | ok |
| 16 | Selected player : 4 | Display ”player number ? “ | Display ”player number ? “ | | ok |
| 17 | Selected player : -5 | Display error message | Display error message | | ok |
| 18 | Return to menu : a | Display error message | Display error message | | ok |
| 19 | End program : 22 | Program should quit | Program crashing error | | bug |
| 20 | Return program : -11 | Display error message | Display error message | | ok |
| 21 | After returning using the menu options : -2 | Display error message | | Display error message | ok |

# Conclusion

* This course work let me to improve my programming and developing skills in,
  + - * + Getting the input from the user and validating it.
        + Saving the inputs in arrays.
        + How to use functional decomposition.
        + How to run the program continuously.
        + How to use method and how to reduce lines in main method.
* Improved on logical expression correctly and how to use it in correct places.
* Using one array, validation and finding more than two winners is more challenging task to write code.
* Things went well while doing this course work,
  + - * Getting input, calculating the winners.
      * Displaying the menu with selection.
      * Displaying error message for invalid inputs (for menu).
      * Adding different function in separate method and calling it again.
* This course work taught me how to do a proper course work in future, analyzing the requirements, planning the structure, designing the program, validating, testing, how to write a report to a course work and time management.
* Experiences and ideas from this course work will help me to do my coursework-two and three better than my first coursework-one.