



## Object Oriented Documentation

alakkarizaki@gmail.com



## Table of Contents

1	Introduction .....	2
2	Software Requirements .....	2
2.1	Class .....	2
2.2	Variables.....	2
2.3	Functions.....	2
2.4	Accessors.....	3
2.5	Attributes .....	3
3	Design Document.....	3
3.1	DoDeductPoints .....	3
3.2	DoAddTeamToLists .....	5
3.3	DoEnterMatchResults .....	6
3.4	DoInitialize .....	9
3.5	DoDisplay .....	11
3.6	DoRelegationZone.....	11
4	Testing requirements .....	12
4.1	Black Box testing. ....	12
4.2	Testing the project .....	12
4.3	Testing for username, password and usertype for leagues.....	12
4.4	Testing for Display and Initialize .....	14
4.5	Testing match results.....	15
4.6	Testing for the deduct points.....	16
4.7	Testing for adding team to list .....	16
4.8	Testing for Do Relegation zone.....	16
5	Conclusion.....	17

## **1 Introduction**

In this documentation it will help us learn and visualise the whole process of the object oriented project in terms of a football league. In this project it will discuss the requirements needed for the program to work. It will also discuss the Designs behind it and the testing of the project of the program.

## **2 Software Requirements**

### **2.1 Class**

This system needs to be set up before the main page. This will have a specific name e.g. for this one it was chosen to be CFootballTeam and CUser. This will be set up into two section which are Private and public. The Private will be what shows up within the headers such as the m\_Name which correlates with the team names. The Public will correlate with the function for each of the options. This will be shown in the Menu. The Public would include the assessors such as DoIntialization. This will eventually be the foundation for how the function will be like.

### **2.2 Variables**

A variable can provide us with named storage that our programs can manipulate. Each variable has their own specific type for C++ programs which can determine the size, range and set of operations that can be applied and stored within the memory and the variable. The names of these variables can be broken down to letters, digits and underscore characters. The type of variables that are essential for this project are int for integer values, bool for true or false determination, char for one byte integer value and void which represents the absence of type. It's basically like math how the variables help out making new and different type of formulas for C++ program projects.

### **2.3 Functions**

A function is a group of statements that can help provide a task. Every C++ program has at least one function, such as main(), and all the other trivial programs can define additional functions. Codes can be divided into separate and different type of functions based on their functionality of which task they perform. The function has two types and they are declaration which tells the compiler about a function's name, return type and parameters. And the other is definition which provides the actual body of the function.

## 2.4 Accessors

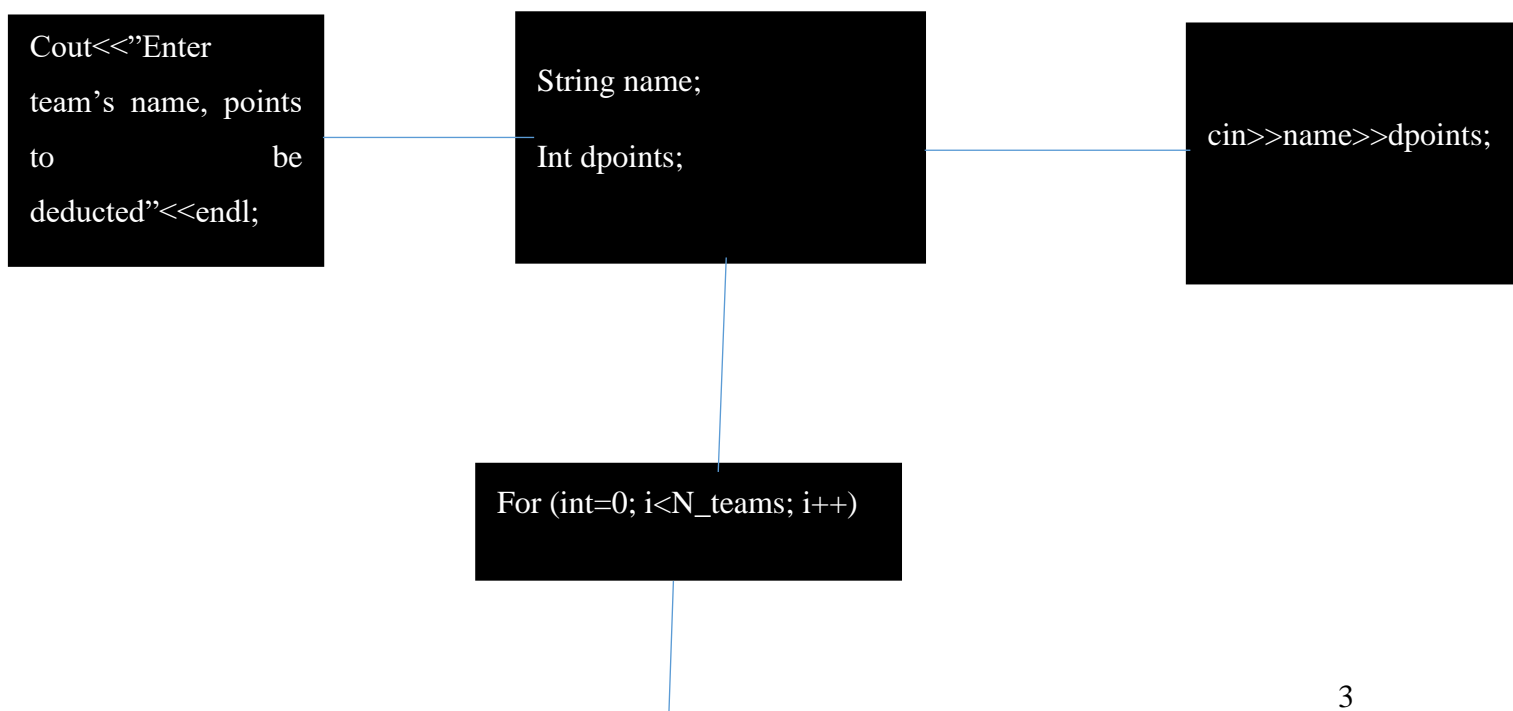
Accessors are another type of functions that are useful except it can only be used it when the programmer set up the classes for object oriented projects. It is the concept of encapsulation. With encapsulation it can help the programmer define labels for the data members and functions and designates whether they are accessible by other classes. When the data members are labelled “private” they cannot be accessed and manipulated by member functions of other classes. Accessors allow access to these private data members. One of the characteristics of the accessors is that it doesn’t need arguments and it has the same type as the retrieved variable. The name of the accessor begins with the Get prefix and the naming convention is necessary.

## 2.5 Attributes

An attribute is a specification that defines a property of an object, element or file. It may also refer to or set the specific value for a given instance of such. Attributes are considered to be meta-data. An attribute is frequently and generally a property of a property. Attribute consists of a name and a value of an element and a type or class name for a file and name and extension. Each named attribute has an associated set of rules called operations. Attributes can be extended with public.

## 3 Design Document

### 3.1 DoDeductPoints



```
If (league[i].HasName(name))  
League[i].DeductPoints(dpoints);
```

```
{  
  
    cout << "Enter team's name,points to be deducted" << endl;  
  
    string name;// using string variables for name  
  
    int dpoints;// using int variables for dpoints  
  
    cin >> name >> dpoints;  
  
  
    for (int i = 0; i < N_teams; i++) //using for loops for the project  
  
    {  
  
        if (league[i].HasName(name))  
  
            league[i].DeductPoints(dpoints);// using the if statement and HasName  
codes and DeductPoints codes  
  
    }  
  
}
```

Basically how the DeductPoints codes are designed by using the cout to make a statement that it wants a team's name and the number of points to be deducted and then we use the cin to put in the answers based on the variables of string and int. Plus we also need to use the for loop in

order for the DeductPoints to work. Then we add in the if statement along with class codes such as HasName to check if the name is true or not and the DeductPoints to subtract the points either 1 or 2.

### 3.2 DoAddTeamToLists



```

{

    string name;//string variable for name

    cout << "Please enter the team's name:"; //cout statement

    cin >> name;//cin to put in the variable

    CFootballTeam team(name);

    if (N_teams < max_N_teams)//if statement codes

    {

        league[N_teams] = team;//codes to save the name of team we entered in the
league table

        N_teams++; //codes to add the team to list

    }

    else

        cout << "Error: you exceded the maximum number of teams" << endl;//cout
statement if there is an error

}

```

To design codes for AddTeamToList we also need to use variables of string and cin and cout statements. If/else statements codes are also used for this function. The DoAddTeamToList function is designed to allow the user to add a new team member. That is also one of the reasons why we use the codes for N\_teams++ and league[N\_teams].

### 3.3 DoEnterMatchResults

```

cout<<" enter match
results in the form of:
hometeam, homescore,
awayteam,
awayscore"<<endl;

```

```

string    hometeam,
awayteam;


int       homescore,
awayscore;

```

```

cin>>hometeam>>homescore>>
awayteam>>awayscore;

```



```
For (int i=0; i<N_teams;i++)
```

```
If (league[i].HasName(hometeam))  
League[i].UpdateOnResult(homescore,awayscore);
```

```
For (int i=0; i<N_teams;i++)
```

```
If (league[i].HasName(awayteam))  
League[i].UpdateOnResult(awayscore,homescore);
```

```
{
```

```
cout << "enter match results in the form: hometeam,homescore,awayteam,awayscore"  
<< endl;//cout statements
```

```
string hometeam, awayteam;//string variable for hometeam, awayteam
```

```
int homescore, awayscore;//int variable for home and away scores
```

```
cin >> hometeam >> homescore >> awayteam >> awayscore;//cin statements
```

```
for (int i = 0; i < N_teams; i++)//for loops
```



```

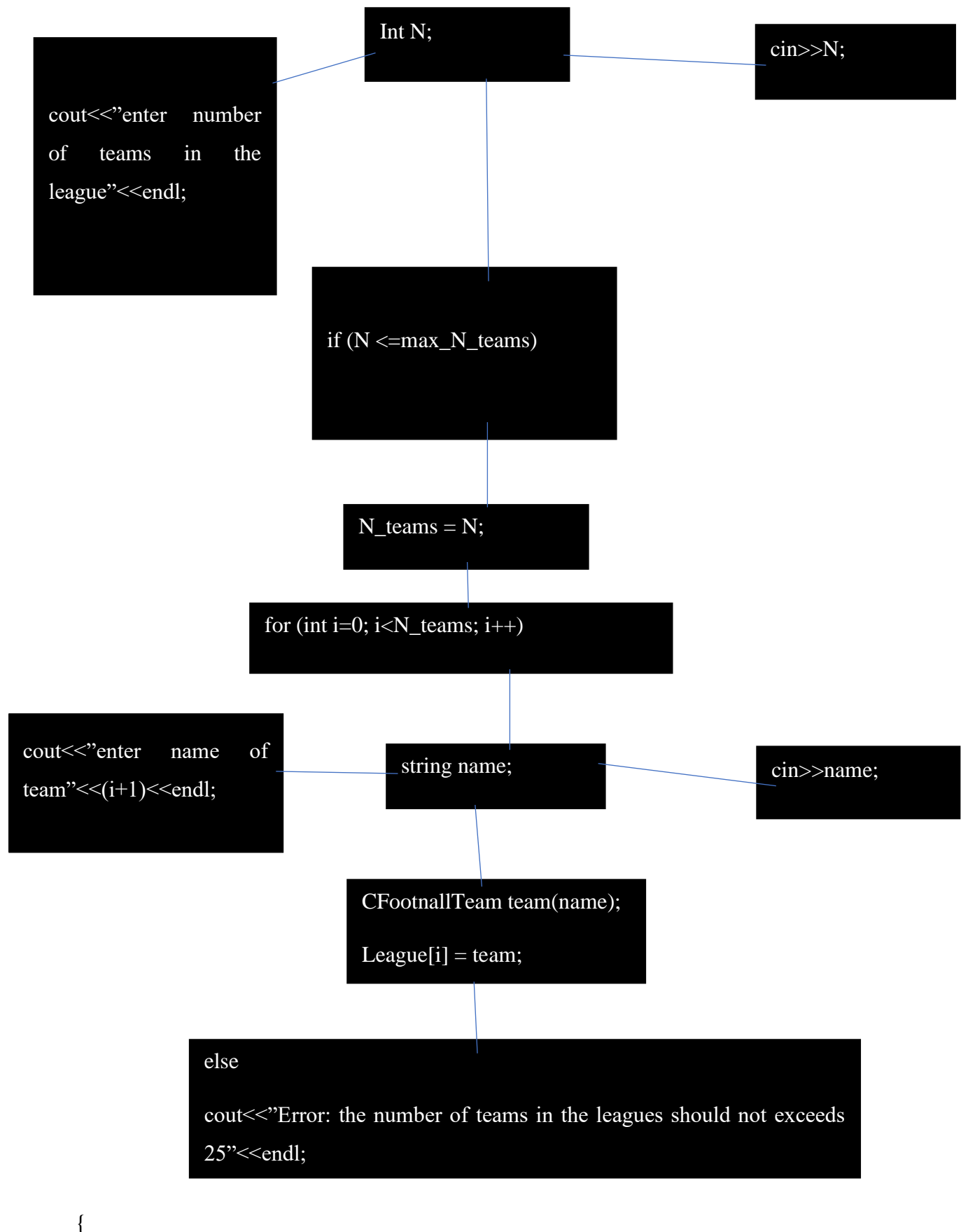
    {
        if (league[i].HasName(hometeam))
            league[i].UpdateOnResult(homescore, awayscore);//if statements for
hometeam
    }

    for (int i = 0; i < N_teams; i++)// another for loop
    {
        if (league[i].HasName(awayteam))
            league[i].UpdateOnResult(awayscore, homescore);//another if
statements for awayteam
    }
}

```

The function DoEnterMatchResult is designed by just using cin/cout statements, variables of string and int, for loops and finally the if statements. This function is carried out by entering the names of hometeam, awayteam and their scores. If both teams tied they get both 1 point but if one of the teams have more points than the other for example the hometeam scored 6 and the awayteam got 3 the hometeam gains 3 points from awayteam.

### 3.4 DoInitialize



```

cout << "enter number of teams in the league" << endl;//cout statements

int N;//int variable for N

cin >> N;//cin statement

if (N <= max_N_teams)//if statement
{
    N_teams = N;

    for (int i = 0; i < N_teams; i++)//for loops
    {
        string name;//string variable for name

        cout << "enter name of team " << (i + 1) << endl;//cout statement

        cin >> name;//cin statement

        CFootballTeam team(name);

        league[i] = team;//using codes from the class CFootballTeam
    }
}

else//else statement
{
    cout << "Error: the number of temas in the league should not exceeds 25" <<
endl;//cout statement
}
}

```

To create codes for DoInitialize we also need to use same as before we use cin/cout statements, variables of int and string, for loops and if/else statements. Plus we need to use class codes from CFootballLeague. How this function is carried out is that it basically asks us to write in

the number of teams we need to add and to name them as well. The number of teams has to be below 25 if it exceeds it then the function will not be carried out.

### 3.5 DoDisplay

```
{  
  
    cout << "Displaying league table" << endl; //cout statement  
  
    cout << " Name " << " GamesPlayed " << " GoalsFor " << " GoalsAgainst " << " Points  
" << endl;  
  
    for (int i = 0; i < N_teams; i++) //for loop  
    {  
        league[i].Display();  
    }  
}
```

This function displays the league table and it only requires few steps.

### 3.6 DoRelegationZone

```
void DoRelegationZone(void)  
{  
    for (int i = N_teams-3 ; i < N_teams; i++)  
        cout << league[i].GetName() << " " << league[i].GetPoints() << " ";  
}
```

*Figure 1 DoRelegationZone Method*

This function is just designed to show the last three teams with their points.

## 4 Testing requirements

### 4.1 Black Box testing.

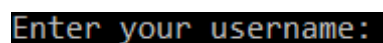
Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

This black box testing will help test the project of how each part of the football league should be lay out. When we test the appearance of the league table we can see as it appease similar to that of the football team.

### 4.2 Testing the project

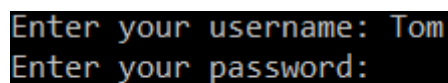
We will now test the project using the black box and then we use several screenshots to explain step by step of how the project works.

### 4.3 Testing for username, password and usertype for leagues



```
Enter your username:
```

When we first opened the black box for the project the first thing it tells us is to enter username which is either the manager Tom or the assistants Fred and Ger.



```
Enter your username: Tom
Enter your password:
```

Now that we entered the name Tom it now asks as us to put in the password and each of the usernames has their own passwords.

```
Enter your username: Tom
Enter your password: 1234
    0.      Initialize League Table
    1.      Display League Table
    2.      Enter a Match Result
    3.      Deduct Points
    4.      Best Defence
    5.      Relegation Zone
    6.      Add Team to list
    7.      Quit
                                Enter Option
```

When we enter the name and password for the manager (name=Tom+ password=1234) they show the application menu for the manager that is related for football league.

```
Enter your username: Fred
Enter your password: 9876
    0.      Display League Table
    1.      Enter a Match Result
    2.      Deduct Points
    3.      Best Defence
    4.      Relegation Zone
    5.      Quit
                                Enter Option
```

But when we enter names and passwords for assistants (names=Fred, Ger +password=9876,4321) they show the application menu for the assistants except the difference is they don't have the Initialise league table and Add team to list in the list that the manager have.

#### 4.4 Testing for Display and Initialize

Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

When testing the Display with either the assistant and the manager it will show us the football league table with empty and zero data. That is because the reason is because the manager hadn't initialized the table yet.

```
enter number of teams in the league
```

When the manager initializes his league tables he has been asked to enter the number of teams in the league.

```
enter number of teams in the league
4
enter name of team 1
a
enter name of team 2
b
enter name of team 3
c
enter name of team 4
d
```

When the manager wants to enter 4 teams in the league he has to enter the names of each team let's just say for example he enters random names of alphabetical letters of a, b, c and d.

Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
a	0	0	0	0
b	0	0	0	0
c	0	0	0	0
d	0	0	0	0

Then when the manager displays his league again it shows the names of the exact teams that the manager initialised. Also, we are going use this league table from now on so that we can test it out with other functions.

#### 4.5 Testing match results

```
enter match results in the form: hometeam,homescore,awayteam,awayscore
a
4
b
4
```

To test for matching results we need the set the names of home team and away team using the names of teams that are initialized such as a and b. Then we enter the scores for both away and home team.

Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
a	0	4	4	1
b	0	4	4	1
c	0	0	0	0
d	0	0	0	0

When we redisplay the league the table has changed because we gained new marks for both GoalsFor and GoalsAgainst between the a and b teams and because they are equal they are bound to gain 1 extra points.

c	0	5	3	3
d	0	3	5	0

Now we set different values for c and d and because c has more marks than d the c gains more marks from the d.



#### 4.6 Testing for the deduct points

```
Enter team's name,points to be deducted
a
2
```

To deduct points from any team we need to enter the team's name first the number of points to be deducted. We chose team a with the deduction points of 2.

Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
a	0	4	4	-1
b	0	4	4	1
c	0	5	3	3
d	0	3	5	0

Then we display the league and then we realize that team a has their points deducted from 1 to -1. This would mean the points can be deducted based on team's name.

#### 4.7 Testing for adding team to list

```
Please enter the team's name:e
```

To add team list to list you just add in the name of the new team you want to add in such as e.

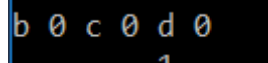
Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
a	0	0	0	0
b	0	0	0	0
c	0	0	0	0
d	0	0	0	0
e	0	0	0	0

Then when we display league you will the new team e has been added to the list.

#### 4.8 Testing for Do Relegation zone

Name	GamesPlayed	GoalsFor	GoalsAgainst	Points
a	0	0	0	0
b	0	0	0	0
c	0	0	0	0
d	0	0	0	0

How to test for Do Relegation we must the option for this function but before we do that we need to initialize the league first.



```
b 0 c 0 d 0
1
```

Then as we finished initializing it, we activate the DoRelegation and as we can see above it shows us the last three teams with their zero points.

## 5 Conclusion

In conclusion what I learned of preparing a project such as the football league is that it is a challenging task. It requires careful mind focus, useful research and a bit of intelligence. It sometimes involves solving a puzzle with a bit of creativity. But it also provides good practice in order to become a successful programmer with a bit of creativity. That is also why the requirements, design and testing documents help us visualise what it also means to be in a world of programming and what it means to work hard. It also provide good practice to find a job relating to programme and codes. That is why it is also important to learn that if you have a dream and goal in life it is important to work hard and fight for it sometimes dreams do come true when putting effort into the work.