

```
1  #include<iostream>
2  #include<string>
3  #include<iomanip>
4  #include<algorithm>
5
6  // OWN LIBRARIES
7  #include "Print.h"
8  #include "upper_char.h"
9  #include "GetLength.h"
10 #include "TeamTable.h"
11
12 using namespace std;
13
14
15 int main() {
16
17     // VARIABLES RELATED TO MENU
18     string MENU_OPTION;
19
20     // ARRAY DECLARATION
21     const size_t SIZE = 12;
22     string TEAM[SIZE];
23     int FOR_GOAL[SIZE], AGAINST_GOAL[SIZE], WIN[SIZE], LOST[SIZE], DRAW [SIZE], POINT[SIZE], MATCH_PLAYED[SIZE];
24
25     // VARIABLE RELATED TO TEAM AND GOAL
26     string HOME_TEAM, VISITING_TEAM;
27     int HOME_GOAL, VISITING_GOAL;
28
29     //VARIABLES RELATED TO INDEXES
30     size_t LENGTH_TEAM, HOME_TEAM_INDEX, VISITING_TEAM_INDEX;
31     size_t TEAM_INDEX = 0;
32
33     // ARRAY INITIALIZATION
34     array_initialize(FOR_GOAL, SIZE);
35     array_initialize(AGAINST_GOAL, SIZE);
36     array_initialize(WIN, SIZE);
37     array_initialize(LOST, SIZE);
38     array_initialize(DRAW, SIZE);
39     array_initialize(POINT, SIZE);
40     array_initialize(MATCH_PLAYED, SIZE);
41
42     do {
43
44         print_menu_option(); // TO PRINT OUT THE AVAILABLE OPTIONS IN MENU
45         cout << "YOUR INPUT:" << setw(5); // ASK USER TO SELECT THE INPUT
46         getline(cin, MENU_OPTION); //SAVE IN THE MENU_OPTION VARIABLE
47         cout << endl << endl;
48
49         MENU_OPTION = string_to_upper(MENU_OPTION); // Change to upper character
50
51         // MENU DRIVEN PROGRAM
```

```
52     if (MENU_OPTION == "ADD") {
53         print_add_banner();
54
55         // TAKE INPUT FOR TEAM NAME
56         cout << " Please Enter the Home Team: ";
57         getline(cin, HOME_TEAM);
58
59         while (get_length(HOME_TEAM) == 0) {
60             cout << " Invalid input! Please Enter the Home Team: ";
61             getline(cin, HOME_TEAM);
62         }
63         cout << " Please Enter the Visiting Team: ";
64         getline(cin, VISITING_TEAM);
65
66         while (get_length(VISITING_TEAM) == 0) {
67             cout << " Invalid input! Please Enter the Visiting Team: ";
68             getline(cin, VISITING_TEAM);
69         }
70
71         // TAKE INPUT FOR TEAM GOAL
72         cout << "How Many goals for home team: ";
73         cin >> HOME_GOAL;
74         cout << "How Many goals for visiting team: ";
75         cin >> VISITING_GOAL;
76
77         cin.ignore();
78
79         // Converting to upper character of Team Name
80         HOME_TEAM = string_to_upper(HOME_TEAM);
81         VISITING_TEAM = string_to_upper(VISITING_TEAM);
82
83         LENGTH_TEAM = get_length(Team);
84
85         if (LENGTH_TEAM == 0) {
86
87             HOME_TEAM_INDEX = TEAM_INDEX;
88             VISITING_TEAM_INDEX = ++TEAM_INDEX;
89
90             //ADD TEAMS NAME IN THE TEAM TABLE
91             add_team_in_table(Team, HOME_TEAM, HOME_TEAM_INDEX);
92             add_team_in_table(Team, VISITING_TEAM, VISITING_TEAM_INDEX);
93         }
94
95         // Otherwise It will check whether team is already existed in
96         // the Table or not. If not then it will add to the table
97         else
98         {
99             HOME_TEAM_INDEX = check_team_in_table(Team, HOME_TEAM,
100             get_length(Team)); //To check whether the Home team is
101             already existed in the table or not
102             VISITING_TEAM_INDEX = check_team_in_table(Team,
103             VISITING_TEAM, get_length(Team)); //To check whether the
```

```
    Visiting team is already existed in the table or not
101
102    // CURRENT INDEX(HOME_TEAM_INDEX OR VISITING_TEAM_INDEX)
    BECOMES ZERO AS THE FUNCTION check_team_in_table
103    // RETURNS 0 WHEN IT DID NOT FIND ANY MATCH TEAM. 0 VALUE
    ALSO CAN BE FOUND WHEN
104    // CONTENT OF THE 0 INDEX IS MATCHED AS WELL.
105
106    if (HOME_TEAM_INDEX == 0 && TEAM[0] != HOME_TEAM) {
107        cout << setw(80) << "NO HOME TEAM MATCH" << endl;
108        HOME_TEAM_INDEX = ++TEAM_INDEX;
109        add_team_in_table(Team, HOME_TEAM, HOME_TEAM_INDEX);
110    }
111
112    if (VISITING_TEAM_INDEX == 0 && TEAM[0] != VISITING_TEAM) {
113        cout << setw(80) << "NO VISITING TEAM MATCH" << endl <<
        endl;
114        VISITING_TEAM_INDEX = ++TEAM_INDEX;
115        add_team_in_table(Team, VISITING_TEAM,
        VISITING_TEAM_INDEX);
116
117    }
118 }
119
120 FOR_GOAL[HOME_TEAM_INDEX] = FOR_GOAL[HOME_TEAM_INDEX] +
    HOME_GOAL;
121 AGAINST_GOAL[HOME_TEAM_INDEX] = AGAINST_GOAL[HOME_TEAM_INDEX] +
    VISITING_GOAL;
122 MATCH_PLAYED[HOME_TEAM_INDEX] = MATCH_PLAYED[HOME_TEAM_INDEX] +
    1;
123
124 FOR_GOAL[VISITING_TEAM_INDEX] = FOR_GOAL[VISITING_TEAM_INDEX] +
    VISITING_GOAL;
125 AGAINST_GOAL[VISITING_TEAM_INDEX] = AGAINST_GOAL
    [VISITING_TEAM_INDEX] + HOME_GOAL;
126 MATCH_PLAYED[VISITING_TEAM_INDEX] = MATCH_PLAYED
    [VISITING_TEAM_INDEX] + 1;
127
128 // CHECK WINNER
129
130 // When Home Team is winner
131 if (HOME_GOAL > VISITING_GOAL) {
132     set_for_win(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX,
        VISITING_TEAM_INDEX);
133 }
134 // When Visiting Team is the winner then goal will be saved in
    For_Goal of visiting team and against goal will save the home
    goal
135 else if (HOME_GOAL < VISITING_GOAL) {
136     set_for_lost(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX,
        VISITING_TEAM_INDEX);
137 }
138 // WHEN MATCH IS DRAW
```

```
139     else if (HOME_GOAL == VISITING_GOAL) {
140         set_for_draw(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX, VISITING_TEAM_INDEX);
141     }
142
143     cout << "PLEASE PRESS ENTER TO RETURN TO MAIN MENU";
144     cin.get();
145 }
146 else if (MENU_OPTION == "PRINT") {
147
148     size_t large_print_space = 20;
149     size_t medium_print_space = 13;
150     size_t small_print_space = 11;
151
152     int MAXIMUM_POINT = 0;
153     int POINT_COPY[SIZE], POINT_SORTED[SIZE];
154
155     size_t INDEX_SORTED[SIZE];
156
157     array_initialize(POINT_COPY, SIZE);
158     array_initialize(POINT_SORTED, SIZE);
159     array_initialize(INDEX_SORTED, SIZE);
160
161     //COPY THE POINT VALUE IN THE POINT_COPY SO THAT MAX FUNCTION COMPARES VALUE>0
162     for (size_t counter0 = 0; counter0 < get_length(Team); counter0++) {
163         int TEMPORARY = 1;
164         POINT_COPY[counter0] = POINT[counter0] + TEMPORARY;
165     }
166
167
168     for (size_t counter1 = 0; counter1 < get_length(Team); counter1++) {
169         INDEX_SORTED[counter1] = find_maxvalue(POINT_COPY, get_length(Team), MAXIMUM_POINT);
170         POINT_SORTED[counter1] = POINT[INDEX_SORTED[counter1]];
171         POINT_COPY[INDEX_SORTED[counter1]] = 0;
172     }
173
174     cout << endl << endl;
175     print_table_coloumn_for_number(medium_print_space);
176     print_table_banner();
177
178     for (size_t TEAM_PRINT_INDEX = 0; TEAM_PRINT_INDEX < get_length(Team); TEAM_PRINT_INDEX++) {
179         print_table_coloumn_for_number(medium_print_space);
180         cout << TEAM[INDEX_SORTED[TEAM_PRINT_INDEX]];
181         print_table_coloumn_for_name(get_length(Team[INDEX_SORTED[TEAM_PRINT_INDEX]]));
182         cout << setw(5) << MATCH_PLAYED[INDEX_SORTED[TEAM_PRINT_INDEX]];
183         print_table_coloumn_for_number(medium_print_space);
```

```
184         cout << setw(5) << FOR_GOAL[INDEX_SORTED[TEAM_PRINT_INDEX]] << " " << AGAINST_GOAL[INDEX_SORTED[TEAM_PRINT_INDEX]];
185         print_table_coloumn_for_number(medium_print_space);
186         cout << setw(3) << WIN[INDEX_SORTED[TEAM_PRINT_INDEX]];
187         print_table_coloumn_for_number(small_print_space);
188         cout << LOST[INDEX_SORTED[TEAM_PRINT_INDEX]];
189         print_table_coloumn_for_number(small_print_space);
190         cout << DRAW[INDEX_SORTED[TEAM_PRINT_INDEX]];
191         print_table_coloumn_for_number(small_print_space);
192         cout << setw(2) << POINT[INDEX_SORTED[TEAM_PRINT_INDEX]] << " " << endl;
193     }
194
195     cout << endl << endl;
196     cout << "PLEASE PRESS ENTER TO RETURN TO MAIN MENU";
197     cin.get();
198
199 }
200 else if (MENU_OPTION == "CLEAR") {
201
202     cout << setw(80) << "TO CLEAN PLEASE ENTER NOW" << endl;
203     cin.get();
204     system("CLS");
205     cout << setw(80) << "PLEASE ENTER AGAIN TO RETURN TO THE MAIN MENU" << endl;
206     cin.get();
207     system("CLS");
208 }
209 else if (MENU_OPTION != "EXIT") {
210     cout << "INVALID INPUT, PLEASE TRY AGAIN THANKS" << endl;
211     cin.clear();
212 }
213 } while (MENU_OPTION != "EXIT");
214
215 cout << endl << endl;
216 system("pause;");
217 return 0;
218 }
```

```
1 #include<iostream>
2 #include<string>
3
4
5 using namespace std;
6
7 //*****
8 // FUNCTION DEFINATION FOR string_to_upper *
9 // THE FUNCTION TAKES A STRING AS AN ARGUMENT AND CONVERTS *
10 // EACH CHARACHTER OF THE STRING INTO UPPER CHARACTER. *
11 // THE FUNCTION RETURNS THE CONVERTED STRING. *
12 //*****
13
14 string string_to_upper(string TEXT) {
15     for (size_t index = 0; index < TEXT.length(); index++) {
16         TEXT[index] = toupper(TEXT[index]); // CALLS THE toupper FUNCTION
17         WITH CHAR INPUT AND RETURNED VALUE IS SAVED IN TEXT[index].
18     }
19     return TEXT;
20 }
```

```

1  #include<iostream>
2  #include<string>
3  #include<iomanip>
4
5  using namespace std;
6
7  //*****
8  // FUNCTION DEFINATION FOR OVERLOAD FUNCTION array_initialize      *
9  // THIS FUNCTION TAKES INTEGER ARRAY AND THE SIZE OF THE ARRAY AS  *
10 // PARAMETER AND INITIALIZE WITH 0 VALUE UNTIL THE SIZE.          *
11 //*****
12 void array_initialize(int ARRAY[], size_t LENGTH) {
13     for (size_t ARRAYINDEX = 0; ARRAYINDEX < LENGTH; ARRAYINDEX++) {
14         ARRAY[ARRAYINDEX] = 0;
15     }
16 }
17
18 //*****
19 // FUNCTION DEFINATION FOR OVERLOAD FUNCTION array_initialize      *
20 // THIS FUNCTION TAKES size_t(unsigned int) ARRAY AND THE SIZE OF  *
21 // THE ARRAY AS FUNCTION PARAMETER AND INITIALIZE WITH 0 VALUE UNTIL  *
22 // THE SIZE.                                                        *
23 //*****
24 void array_initialize(size_t ARRAY[], size_t LENGTH) {
25     for (size_t ARRAYINDEX = 0; ARRAYINDEX < LENGTH; ARRAYINDEX++) {
26         ARRAY[ARRAYINDEX] = 0;
27     }
28 }
29
30 //*****
31 // FUNCTION DEFINATION FOR add_team_in_table                        *
32 // THIS FUNCTION TAKES STRING ARRAY, STRING AND THE INDEX OF THE ARRAY *
33 // WHERE THE TEAM NAME WILL BE SAVED.                               *
34 //*****
35 void add_team_in_table(string TEAM_TABLE[], string TEAM_NAME, size_t INDEX) {
36     {
37         TEAM_TABLE[INDEX] = TEAM_NAME;
38         cout << setw(80) << "THANKS! NEW TEAM IS ADDED TO THE TABLE" << endl <<
39         endl;
40     }
41 //
42 //*****
43 // FUNCTION DEFINATION FOR check_team_in_table                      *
44 // THIS FUNCTION TAKES STRING ARRAY, STRING AND LENGTH OF THE ARRAY AS *
45 // INPUT ARGUMENT TO CHECK WHETHER THE TEAM IS ALREADY IN THE TABLE OR *
46 // NOT. IF THE TEAM NAME MATCHES WITH THE EXISTING TEAM IN THE TABLE, IT *
47 // RETURNS THE INDEX OF MATCED TEAM. IF TEAM DOES NOT EXIST IN THE TABLE IT

```

```

*
46 // RETURNS 0.
*
47 //
*****
*
48 size_t check_team_in_table(string TEAM_TABLE[], string TEAM_NAME, size_t
    LENGTH) {
49
50     size_t index_to_return = 0;
51     for (size_t counter0 = 0; counter0 < LENGTH; counter0++) {
52
53         if (TEAM_NAME == TEAM_TABLE[counter0]) {
54             cout << setw(80) << "ENTERED TEAM IS ALREADY IN THE TABLE" <<
                endl << endl;
55             index_to_return = counter0;
56         }
57     }
58     //cout << index_to_return << endl;
59     return index_to_return;
60 }
61
62 //
*****
*
63 // FUNCTION DEFINATION FOR set_for_win
*
64 // THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS
*
65 // INPUT ARGUMENT TO SET THE FOLOWWING TABLE WHEN THE TEAM HAS OWN A MATCH.
*
66 //
*****
*
67 void set_for_win(int WIN[], int LOST[], int DRAW[], int POINT[], size_t
    INDEX1, size_t INDEX2) {
68
69     WIN[INDEX1] = WIN[INDEX1] + 1;
70     LOST[INDEX1] = LOST[INDEX1] + 0;
71     DRAW[INDEX1] = DRAW[INDEX1] + 0;
72     POINT[INDEX1] = POINT[INDEX1] + 3;
73
74     WIN[INDEX2] = WIN[INDEX2] + 0;
75     LOST[INDEX2] = LOST[INDEX2] + 1;
76     DRAW[INDEX2] = DRAW[INDEX2] + 0;
77     POINT[INDEX2] = POINT[INDEX2] + 0;
78 }
79
80 //
*****
*
81 // FUNCTION DEFINATION FOR set_for_lost
*

```



```

...ssignment\Final_Assignment\Final_Assignment\TeamTable.h 3
82 // THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS  ↗
    *
83 // INPUT ARGUMENT TO SET THE FOLOWING TABLE WHEN THE TEAM HAS LOST A  ↗
    MATCH.*
84 //  ↗
    *****  ↗
    *
85 void set_for_lost(int WIN[], int LOST[], int DRAW[], int POINT[], size_t  ↗
    INDEX1, size_t INDEX2) {
86
87     WIN[INDEX1] = WIN[INDEX1] + 0;
88     LOST[INDEX1] = LOST[INDEX1] + 1;
89     DRAW[INDEX1] = DRAW[INDEX1] + 0;
90     POINT[INDEX1] = POINT[INDEX1] + 0;
91
92     WIN[INDEX2] = WIN[INDEX2] + 1;
93     LOST[INDEX2] = LOST[INDEX2] + 0;
94     DRAW[INDEX2] = DRAW[INDEX2] + 0;
95     POINT[INDEX2] = POINT[INDEX2] + 3;
96 }
97
98 //  ↗
    *****  ↗
    *****
99 // FUNCTION DEFINATION FOR set_for_draw  ↗
    *
100 // THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS  ↗
    *
101 // INPUT ARGUMENT TO SET THE FOLLOWING TABLE WHEN THE TEAMs HAS DRAWN A  ↗
    MATCH. *
102 //  ↗
    *****  ↗
    *****
103 void set_for_draw(int WIN[], int LOST[], int DRAW[], int POINT[], size_t  ↗
    INDEX1, size_t INDEX2) {
104
105     WIN[INDEX1] = WIN[INDEX1] + 0;
106     LOST[INDEX1] = LOST[INDEX1] + 0;
107     DRAW[INDEX1] = DRAW[INDEX1] + 1;
108     POINT[INDEX1] = POINT[INDEX1] + 1;
109
110     WIN[INDEX2] = WIN[INDEX2] + 0;
111     LOST[INDEX2] = LOST[INDEX2] + 0;
112     DRAW[INDEX2] = DRAW[INDEX2] + 1;
113     POINT[INDEX2] = POINT[INDEX2] + 1;
114 }
115
116 //  ↗
    *****  ↗
    *****
117 // FUNCTION DEFINATION FOR find_maxvalue  ↗
    *
118 // THIS FUNCTION TAKES INT ARRAY, LENGTH OF THE ARRAY AND MAXIMUM POINT AS  ↗

```

```

    *
119 // THE INPUT ARGUMENT AND RETURNS THE INDEX OF THE MAXIMUM VALUE OF THE  ↗
    ARRAY.  *
120 //                                  ↗
    ***** ↗
    *****
121 size_t find_maxvalue(int POINTCOPY[], size_t LENGTH, int MAXIMUM_POINT) {
122
123     size_t index0 = 0;
124     for (size_t counter1 = 0; counter1 < LENGTH; counter1++) {
125         if (POINTCOPY[counter1] > MAXIMUM_POINT) {
126             MAXIMUM_POINT = POINTCOPY[counter1];
127             index0 = counter1;
128         }
129     }
130     return index0;
131 }
```

```
1  #include <iostream>
2  #include<string>
3
4  using namespace std;
5
6  /*******
7  // FUNCTION DEFINATION FOR OVERLOAD FUNCTION get_length      *
8  // THIS FUNCTION USES AN STRING PARAMETER AND                *
9  // RETURNS THE LENGTH AS SIZE_T                               *
10 /*******
11
12 size_t get_length(string STRING) {
13
14     size_t LENGTH = STRING.length();
15     return LENGTH;
16 }
17
18
19 /*******
20 // FUNCTION DEFINATION FOR OVERLOAD FUNCTION get_length      *
21 // THIS FUNCTION USES AN STRING ARRAY AS PARAMETER AND       *
22 // RETURNS THE LENGTH AS SIZE_T                               *
23 /*******
24
25 size_t get_length(string STRINGARRAY[]) {
26     size_t LENGTH_ARRAY = 0;
27     while (!STRINGARRAY[LENGTH_ARRAY].empty()) {
28         ++LENGTH_ARRAY;
29     }
30     return LENGTH_ARRAY;
31 }
32
33
```

```

1  #include<iostream>
2  #include<iomanip>
3
4
5  using namespace std;
6
7
8  //
   *****
   **
9  // FUNCTION DEFINATION FOR print_menu_option
   *
10 // THE FUNCTION PRINTS THE FOLLOWING AVAILABLE OPTIONS IN THE MENU
   *
11 // ADD: TO ADD MATCH INFORMATION SUCH AS ADD TEAM RECORDS, UPDATE POINT TABLE
   *
12 // PRINT: TO PRINT OUT THE CURRENT TEAM STANDING
   *
13 // CLEAR: TO CLEAR THE SCREEN
   *
14 // EXIT: TO EXIT FROM THE PROGRAM
   *
15 //
   *****
   **
16 void print_menu_option() {
17
18     // Get the Menu option from the user
19     cout << setw(85) << "PLEASE ENTER YOUR CHOICE FROM THE BELOW OPTIONS \n"
        << endl;
20     cout << endl;
21     cout << setw(100) <<
        "
        _____\n";
22     cout << setw(94) << "|OPTIONS      |              DESCRIPTIONS"
        |\n";
23     cout << setw(101) << "|*****|"
        "*****| \n";
24     cout << setw(70) << "|ADD          |   TO ADD SCORE OF THE NEW MATCH."
        |\n";
25     cout << setw(87) << "|PRINT        |   TO PRINT THE STANDING TABLE OF THE
        CHAMPIONSHIP. |\n";
26     cout << setw(63) << "|CLEAR        |   TO CLEAR THE SCREEN."
        |\n";
27     cout << setw(66) << "|EXIT         |   TO EXIT FROM THE PROGRAM."
        |\n";
28     cout << setw(99) << "|_____|"
        "_____|" ;
29
30     cout << endl << endl;
31
32 }
33

```

```

...ng Assignment\Final_Assignment\Final_Assignment\Print.h 2
34 // 2
    ***** 2
    **
35 // FUNCTION DEFINATION FOR print_add_banner 2
    *
36 // THE FUNCTION PRINTS THE FOLLOWING MESSAGE IN THE PRINTING MENU 2
    *
37 // 2
    ***** 2
    **
38 void print_add_banner() {
39     cout << setw(81) << "!!!WELCOME TO THE CHAMPIONSHIP!!!" << endl;
40     cout << setw(80) << "PLEASE ENTER THE FOLLOWING INFROMATION" << endl;
41 }
42
43 // 2
    ***** 2
    *****
44 // FUNCTION DEFINATION FOR print_table_coloumn_for_name 2
    *
45 // THE FUNCTION PRINTS WHITESPACE ACCORDING TO THE size_t SPACE VARIABLE 2
    SPECIFIED *
46 // AS FUNCTION INPUT TARGUMENT TO PROVIDE APROPRIATE SPACE DEPENDING ON THE 2
    NAME OF THE TEAM *
47 // 2
    ***** 2
    *****
48 void print_table_coloumn_for_name(size_t SPACE) {
49
50     size_t possible_highest_team_name = 20;
51     for (size_t counter = 0; counter < possible_highest_team_name - SPACE; 2
        counter++) {
52         cout << " ";
53     }
54 }
55
56 // 2
    ***** 2
    *****
57 // FUNCTION DEFINATION FOR print_table_coloumn_for_number 2
    *
58 // THE FUNCTION PRINTS SPACE IN THE TEAM STANDING TABLE IN THE PRINT MENU. 2
    *
59 // THIS FUNCTION PRINTS WHITESPACE ACCORDING TO THE NUMBER SPECIFIEDBY size_t 2
    SPACE VARIABLE *
60 // 2
    ***** 2
    *****
61 void print_table_coloumn_for_number(size_t SPACE) {
62     for (size_t counter = 0; counter < SPACE; counter++) {
63         cout << " ";
64     }
65 }

```

```

66
67 //
        *****
        *****
68 // FUNCTION DEFINATION FOR print_table_coloumn_for_name
        *
69 // THE FUNCTION PRINTS THE FOLLOWING BANNER FOR THE TEAM STANDING IN THE
    PRINT MENU      *
70 //
        *****
        *****
71 void print_table_banner() {
72
73     size_t print_name = 8;
74     size_t print_number = 12;
75     cout << "TEAM";
76     print_table_coloumn_for_name(print_name);
77     cout << "Match Played";
78     print_table_coloumn_for_name(print_number);
79     cout << "FOR-AGAINST";
80     print_table_coloumn_for_name(print_number);
81     cout << "WIN";
82     print_table_coloumn_for_name(print_number);
83     cout << "LOST";
84     print_table_coloumn_for_name(print_number);
85     cout << "DRAW";
86     print_table_coloumn_for_name(print_number);
87     cout << "POINTS" << endl;
88     print_table_coloumn_for_name(print_number);
89     cout <<
        *****
        *****\n" << endl;
90 }
91
92

```