```
1 #include<iostream>
 2 #include<string>
 3 #include<iomanip>
 4 #include<algorithm>
 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
       // VARIABLES RELATED TO MENU
17
18
       string MENU_OPTION;
19
20
       // ARRAY DECLARATION
       const size_t SIZE = 12;
21
22
       string TEAM[SIZE];
       int FOR_GOAL[SIZE], AGAINST_GOAL[SIZE], WIN[SIZE], LOST[SIZE], DRAW
23
          [SIZE], POINT[SIZE], MATCH_PLAYED[SIZE];
24
25
       // VARIABLE RELATED TO TEAM AND GOAL
26
       string HOME TEAM, VISITING TEAM;
       int HOME_GOAL, VISITING_GOAL;
27
28
       //VARIABLES RELATED TO INDEXS
29
30
       size_t LENGTH_TEAM, HOME_TEAM_INDEX, VISITING_TEAM_INDEX;
       size_t TEAM_INDEX = 0;
31
32
33
       // ARRAY INITIALIZATION
34
       array_initialize(FOR_GOAL, SIZE);
35
       array_initialize(AGAINST_GOAL, SIZE);
       array_initialize(WIN, SIZE);
36
       array initialize(LOST, SIZE);
37
38
       array_initialize(DRAW, SIZE);
39
       array_initialize(POINT, SIZE);
       array_initialize(MATCH_PLAYED, SIZE);
40
41
       do {
42
43
            print_menu_option(); // TO PRINT OUT THE AVAILABLE OPTIONS IN MENU
44
            cout << "YOUR INPUT:" << setw(5); // ASK USER TO SELECT THE INPUT</pre>
45
46
            getline(cin, MENU_OPTION); //SAVE IN THE MENU_OPTION VARIABLE
47
            cout << endl << endl;</pre>
48
49
           MENU_OPTION = string_to_upper(MENU_OPTION); // Change to upper
              character
50
51
            // MENU DRIVEN PROGRAM
```

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

VISITING_TEAM, get_length(TEAM)); //To check whether the

```
\dotsg Assignment\Final_Assignment\Final_Assignment\main.cpp
```

```
3
```

```
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
                     // CONTENT OF THE 0 INDEX IS MATCHED AS WELL.
104
105
                     if (HOME TEAM INDEX == 0 && TEAM[0] != HOME TEAM) {
106
                         cout << setw(80) << "NO HOME TEAM MATCH" << endl;</pre>
107
108
                         HOME_TEAM_INDEX = ++TEAM_INDEX;
                         add_team_in_table(TEAM, HOME_TEAM, HOME_TEAM_INDEX);
109
                     }
110
111
                     if (VISITING_TEAM_INDEX == 0 && TEAM[0] != VISITING_TEAM) {
112
                         cout << setw(80) << "NO VISITING TEAM MATCH" << endl << →
113
                         endl;
                         VISITING TEAM INDEX = ++TEAM INDEX;
114
                         add team in table(TEAM, VISITING TEAM,
115
                                                                                    P
                        VISITING_TEAM_INDEX);
116
117
                     }
                 }
118
119
120
                 FOR GOAL[HOME TEAM INDEX] = FOR GOAL[HOME TEAM INDEX] +
                   HOME GOAL;
                 AGAINST GOAL[HOME TEAM INDEX] = AGAINST GOAL[HOME TEAM INDEX] +
121
                   VISITING GOAL;
122
                 MATCH PLAYED[HOME TEAM INDEX] = MATCH PLAYED[HOME TEAM INDEX] + >
123
                 FOR GOAL[VISITING TEAM INDEX] = FOR GOAL[VISITING TEAM INDEX] +
124
                   VISITING GOAL;
                 AGAINST_GOAL[VISITING_TEAM_INDEX] = AGAINST_GOAL
125
                   [VISITING_TEAM_INDEX] + HOME_GOAL;
                 MATCH_PLAYED[VISITING_TEAM_INDEX] = MATCH_PLAYED
126
                   [VISITING TEAM INDEX] + 1;
127
                 // CHECK WINNER
128
129
                 // When Home Team is winner
130
                 if (HOME GOAL > VISITING GOAL) {
131
                     set_for_win(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX,
132
                       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
                 else if (HOME GOAL < VISITING GOAL) {</pre>
135
136
                     set_for_lost(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX,
                       VISITING TEAM INDEX);
137
138
                 // WHEN MATCH IS DRAW
```

```
...g Assignment\Final_Assignment\Final_Assignment\main.cpp
139
                 else if (HOME GOAL == VISITING GOAL) {
140
                      set_for_draw(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX,
                       VISITING_TEAM_INDEX);
                 }
141
142
                 cout << "PLEASE PRESS ENTER TO RETURN TO MAIN MENU";</pre>
143
144
                 cin.get();
             }
145
             else if (MENU OPTION == "PRINT") {
146
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);
                 array_initialize(POINT_SORTED, SIZE);
158
159
                 array_initialize(INDEX_SORTED, SIZE);
160
                 //COPY THE POINT VALUE IN THE POINT_COPY SO THAT MAX FUNCTION
161
                   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]];
                     POINT COPY[INDEX SORTED[counter1]] = 0;
171
                 }
172
173
174
                 cout << endl << endl;</pre>
175
                 print_table_coloumn_for_number(medium_print_space);
176
                 print_table_banner();
177
                 for (size_t TEAM_PRINT_INDEX = 0; TEAM_PRINT_INDEX < get_length >
178
                   (TEAM); TEAM_PRINT_INDEX++) {
179
                     print_table_coloumn_for_number(medium_print_space);
180
                     cout << TEAM[INDEX_SORTED[TEAM_PRINT_INDEX]];</pre>
181
                     print_table_coloumn_for_name(get_length(TEAM[INDEX_SORTED
                        [TEAM_PRINT_INDEX]]));
182
                      cout << setw(5) << MATCH PLAYED[INDEX SORTED</pre>
                        [TEAM_PRINT_INDEX]];
183
                     print_table_coloumn_for_number(medium_print_space);
```

```
...g Assignment\Final_Assignment\Final_Assignment\main.cpp
```

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

```
1 #include<iostream>
2 #include<string>
3
4
5 using namespace std;
6
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.
13
14 string string_to_upper(string TEXT) {
      for (size_t index = 0; index < TEXT.length(); index++) {</pre>
15
         TEXT[index] = toupper(TEXT[index]); // CALLS THE toupper FUNCTION
          WITH CHAR INPUT AND RETURNED VALUE IS SAVED IN TEXT[index].
17
      }
      return TEXT;
18
19 }
20
```

```
#include<iostream>
2 #include<string>
3 #include<iomanip>
4
5 using namespace std;
6
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.
12 void array_initialize(int ARRAY[], size_t LENGTH) {
      for (size_t ARRAYINDEX = 0; ARRAYINDEX < LENGTH; ARRAYINDEX++) {</pre>
13
14
         ARRAY[ARRAYINDEX] = 0;
15
16 }
17
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.
void array_initialize(size_t ARRAY[], size_t LENGTH) {
25
      for (size t ARRAYINDEX = 0; ARRAYINDEX < LENGTH; ARRAYINDEX++) {</pre>
         ARRAY[ARRAYINDEX] = 0;
26
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.
35 void add_team_in_table(string TEAM_TABLE[], string TEAM_NAME, size_t INDEX) →
      TEAM TABLE[INDEX] = TEAM NAME;
36
      cout << setw(80) << "THANKS! NEW TEAM IS ADDED TO THE TABLE" << endl << ▷
37
       endl;
38 }
39
  // FUNCTION DEFINATION FOR check_team_in_table
  // THIS FUNCTION TAKES STRING ARRAY, STRING AND LENGTH OF THE ARRAY AS
  // INPUT ARGUMENT TO CHECK WHETHER THE TEAM IS ALREADY IN THE TABLE OR
  // NOT. IF THE TEAM NAME MATCHES WITH THE EXISTING TEAM IN THE TABLE, IT
45 // RETURNS THE INDEX OF MATCED TEAM. IF TEAM DOES NOT EXIST IN THE TABLE IT >
```

80 //

81 // FUNCTION DEFINATION FOR set_for_lost

```
...ssignment\Final_Assignment\TeamTable.h
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;
       for (size_t counter0 = 0; counter0 < LENGTH; counter0++) {</pre>
51
52
           if (TEAM_NAME == TEAM_TABLE[counter0]) {
53
              cout << setw(80) << "ENTERED TEAM IS ALREADY IN THE TABLE" << →
                endl << endl;</pre>
55
              index to return = counter0;
56
           }
57
58
       //cout << index_to_return << endl;</pre>
59
       return index to return;
60 }
61
62 //
     *************************
63 // FUNCTION DEFINATION FOR set for win
   // 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
       WIN[INDEX1] = WIN[INDEX1] + 1;
69
70
       LOST[INDEX1] = LOST[INDEX1] + 0;
71
       DRAW[INDEX1] = DRAW[INDEX1] + 0;
       POINT[INDEX1] = POINT[INDEX1] + 3;
72
73
74
       WIN[INDEX2] = WIN[INDEX2] + 0;
75
       LOST[INDEX2] = LOST[INDEX2] + 1;
76
       DRAW[INDEX2] = DRAW[INDEX2] + 0;
       POINT[INDEX2] = POINT[INDEX2] + 0;
77
78 }
79
```

```
\dots s signment \verb|\Final_Assignment| Team Table. h
 82 // THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS
 83 // INPUT ARGUMENT TO SET THE FOLOWWING TABLE WHEN THE TEAM HAS LOST A
 84 //
      ***************************
 85 void set_for_lost(int WIN[], int LOST[], int DRAW[], int POINT[], size_t
      INDEX1, size_t INDEX2) {
 86
        WIN[INDEX1] = WIN[INDEX1] + 0;
 87
        LOST[INDEX1] = LOST[INDEX1] + 1;
 88
 89
        DRAW[INDEX1] = DRAW[INDEX1] + 0;
        POINT[INDEX1] = POINT[INDEX1] + 0;
 90
 91
        WIN[INDEX2] = WIN[INDEX2] + 1;
 92
        LOST[INDEX2] = LOST[INDEX2] + 0;
 93
        DRAW[INDEX2] = DRAW[INDEX2] + 0;
        POINT[INDEX2] = POINT[INDEX2] + 3;
95
 96 }
 97
      *****************************
 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
102 //
103 void set_for_draw(int WIN[], int LOST[], int DRAW[], int POINT[], size_t
      INDEX1, size_t INDEX2) {
104
        WIN[INDEX1] = WIN[INDEX1] + 0;
105
        LOST[INDEX1] = LOST[INDEX1] + 0;
106
107
        DRAW[INDEX1] = DRAW[INDEX1] + 1;
108
        POINT[INDEX1] = POINT[INDEX1] + 1;
109
110
        WIN[INDEX2] = WIN[INDEX2] + 0;
        LOST[INDEX2] = LOST[INDEX2] + 0;
111
        DRAW[INDEX2] = DRAW[INDEX2] + 1;
112
        POINT[INDEX2] = POINT[INDEX2] + 1;
113
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++) {</pre>
125
          if (POINTCOPY[counter1] > MAXIMUM_POINT) {
              MAXIMUM_POINT = POINTCOPY[counter1];
126
127
              index0 = counter1;
128
          }
129
       }
       return index0;
130
131 }
```

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

```
...ng Assignment\Final_Assignment\Final_Assignment\Print.h
```

```
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
                                                                      P
13 // CLEAR: TO CLEAR THE SCREEN
14 // EXIT: TO EXIT FROM THE PROGRAM
15 //
    ***********************
16 void print_menu_option() {
17
      // Get the Menu option from the user
18
      cout << setw(85) << "PLEASE ENTER YOUR CHOICE FROM THE BELOW OPTIONS \n" →
        << endl;
20
      cout << endl;</pre>
      cout << setw(100) <<
21
                                                                      P
                                                                      P
          __\n";
22
      cout << setw(94) << "OPTIONS
                                                     DESCRIPTIONS
                      \n";
      cout << setw(101) << "|*********
23
        *********** \n";
      cout << setw(70) << "|ADD | TO ADD SCORE OF THE NEW MATCH.
24
             \n";
      cout << setw(87) << "|PRINT | TO PRINT THE STANDING TABLE OF THE
25
        CHAMPIONSHIP. |\n";
      cout << setw(63) << "|CLEAR | TO CLEAR THE SCREEN.</pre>
26
                                                                      P
         \n";
      cout << setw(66) << "|EXIT | TO EXIT FROM THE PROGRAM.</pre>
27
         \n";
      cout << setw(99) << "
28
29
30
      cout << endl << endl;</pre>
31
32 }
33
```

```
...ng Assignment\Final_Assignment\Final_Assignment\Print.h
         *************************
35 // FUNCTION DEFINATION FOR print_add_banner
36 // THE FUNCTION PRINTS THE FOLLOWING MESSAGE IN THE PRINTING MENU
37 //
     *************************
38 void print_add_banner() {
      cout << setw(81) << "!!!WELCOME TO THE CHAMPIONSHIP!!!" << endl;</pre>
40
      cout << setw(80) << "PLEASE ENTER THE FOLLOWING INFROMATION" << endl;</pre>
41 }
42
43 //
                    **************
44 // FUNCTION DEFINATION FOR print_table_coloumn_for_name
45 // THE FUNCTION PRINTS WHITESPACE ACCORDING TO THE size t SPACE VARIABLE
     SPECIFIED
46 // AS FUNCTION INPUT TARGUMENT TO PROVIDE APROPRIATE SPACE DEPENDING ON THE >
     NAME OF THE TEAM *
47 //
     ***************************
48 void print_table_coloumn_for_name(size_t SPACE) {
49
50
      size_t possible_highest_team_name = 20;
      for (size t counter = 0; counter < possible highest team name - SPACE;</pre>
        counter++) {
          cout << " ";
52
53
54 }
55
56 //
57 // FUNCTION DEFINATION FOR print table coloumn for number
58 // THE FUNCTION PRINTS SPACE IN THE TEAM STANDING TABLE IN THE PRINT MENU.
59 // THIS FUNCTION PRINTS WHITESPACE ACCORDING TO THE NUMBER SPECIFIEDBY size t >
     SPACE VARIABLE *
60 //
     *************************
61 void print table coloumn for number(size t SPACE) {
62
      for (size_t counter = 0; counter < SPACE; counter++) {</pre>
          cout << " ";
63
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;
      cout << "TEAM";</pre>
75
76
      print_table_coloumn_for_name(print_name);
77
      cout << "Match Played";</pre>
78
      print_table_coloumn_for_name(print_number);
79
      cout << "FOR-AGAINST";</pre>
80
      print_table_coloumn_for_name(print_number);
      cout << "WIN";</pre>
81
82
      print_table_coloumn_for_name(print_number);
      cout << "LOST";</pre>
83
84
      print table coloumn for name(print number);
85
      cout << "DRAW";</pre>
      print_table_coloumn_for_name(print_number);
86
      cout << "POINTS" << endl;</pre>
87
88
      print_table_coloumn_for_name(print_number);
89
        **********************************\n" << endl;
90 }
91
92
```

...ng Assignment\Final_Assignment\Final_Assignment\Print.h