```
#include<iostream>
#include<string>
#include<iomanip>
#include<algorithm>

// OWN LIBRARIES
#include "Print.h"
#include "upper_char.h"
#include "GetLength.h"
#include "TeamTable.h"

using namespace std;


int main() {

    // VARIABLES RELATED TO MENU
    string MENU_OPTION;

    // ARRAY DECLARATION
    const size_t SIZE = 12;
    string TEAM[SIZE];
    int FOR_GOAL[SIZE], AGAINST_GOAL[SIZE], WIN[SIZE], LOST[SIZE], DRAW[SIZE], POINT[SIZE], MATCH_PLAYED[SIZE];

    // VARIABLE RELATED TO TEAM AND GOAL
    string HOME_TEAM, VISITING_TEAM;
    int HOME_GOAL, VISITING_GOAL;

    //VARIABLES RELATED TO INDEXS
    size_t LENGTH_TEAM, HOME_TEAM_INDEX, VISITING_TEAM_INDEX;
    size_t TEAM_INDEX = 0;

    // ARRAY INITIALIZATION
    array_initialize(FOR_GOAL, SIZE);
    array_initialize(AGAINST_GOAL, SIZE);
    array_initialize(WIN, SIZE);
    array_initialize(LOST, SIZE);
    array_initialize(DRAW, SIZE);
    array_initialize(POINT, SIZE);
    array_initialize(MATCH_PLAYED, SIZE);

    do {

        print_menu_option(); // TO PRINT OUT THE AVAILABLE OPTIONS IN MENU
        cout << "YOUR INPUT:" << setw(5); // ASK USER TO SELECT THE INPUT
        getline(cin, MENU_OPTION); //SAVE IN THE MENU_OPTION VARIABLE
        cout << endl << endl;

        MENU_OPTION = string_to_upper(MENU_OPTION);  // Change to upper character

        // MENU DRIVEN PROGRAM
        if (MENU_OPTION == "ADD") {
            print_add_banner();

            // TAKE INPUT FOR TEAM NAME
            cout << " Please Enter the Home Team: ";
            getline(cin, HOME_TEAM);

            while (get_length(HOME_TEAM) == 0) {
                cout << " Invalid input! Please Enter the Home Team: ";
                getline(cin, HOME_TEAM);
            }
            cout << " Please Enter the Visiting Team: ";
            getline(cin, VISITING_TEAM);

            while (get_length(VISITING_TEAM) == 0) {
                cout << " Invalid input! Please Enter the Visiting Team: ";
                getline(cin, VISITING_TEAM);
            }

            // TAKE INPUT FOR TEAM GOAL
            cout << "How Many goals for home team: ";
            cin >> HOME_GOAL;
            cout << "How Many goals for visiting team: ";
            cin >> VISITING_GOAL;

            cin.ignore();

            // Converting to upper character of Team Name
            HOME_TEAM = string_to_upper(HOME_TEAM);
            VISITING_TEAM = string_to_upper(VISITING_TEAM);

            LENGTH_TEAM = get_length(TEAM);

            if (LENGTH_TEAM == 0) {

                HOME_TEAM_INDEX = TEAM_INDEX;
                VISITING_TEAM_INDEX = ++TEAM_INDEX;

                //ADD TEAMS NAME IN THE TEAM TABLE
                add_team_in_table(TEAM, HOME_TEAM, HOME_TEAM_INDEX);
                add_team_in_table(TEAM, VISITING_TEAM, VISITING_TEAM_INDEX);
            }

            // Otherwise It will check whether team is alsready existed in
              the Table or not. If not then it will add to the table
            else
            {

                HOME_TEAM_INDEX = check_team_in_table(TEAM, HOME_TEAM,
                  get_length(TEAM)); //To check whether the Home team is
                  already existed in the table or not
                VISITING_TEAM_INDEX = check_team_in_table(TEAM,
                  VISITING_TEAM, get_length(TEAM));//To check whether the
                  Visiting team is already existed in the table or not

                // CURRENT INDEX(HOME_TEAM_INDEX OR VISITING_TEAM_INDEX) BECOMES ZERO AS THE FUNCTION check_team_in_table
                // 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.

                if (HOME_TEAM_INDEX == 0 && TEAM[0] != HOME_TEAM) {
                    cout << setw(80) << "NO HOME TEAM MATCH" << endl;
                    HOME_TEAM_INDEX = ++TEAM_INDEX;
                    add_team_in_table(TEAM, HOME_TEAM, HOME_TEAM_INDEX);
                }

                if (VISITING_TEAM_INDEX == 0 && TEAM[0] != VISITING_TEAM) {
                    cout << setw(80) << "NO VISITING TEAM MATCH" << endl << endl;
                    VISITING_TEAM_INDEX = ++TEAM_INDEX;
                    add_team_in_table(TEAM, VISITING_TEAM, VISITING_TEAM_INDEX);

                }
            }

            FOR_GOAL[HOME_TEAM_INDEX] = FOR_GOAL[HOME_TEAM_INDEX] + HOME_GOAL;
            AGAINST_GOAL[HOME_TEAM_INDEX] = AGAINST_GOAL[HOME_TEAM_INDEX] + VISITING_GOAL;
            MATCH_PLAYED[HOME_TEAM_INDEX] = MATCH_PLAYED[HOME_TEAM_INDEX] + 1;

            FOR_GOAL[VISITING_TEAM_INDEX] = FOR_GOAL[VISITING_TEAM_INDEX] + VISITING_GOAL;
            AGAINST_GOAL[VISITING_TEAM_INDEX] = AGAINST_GOAL[VISITING_TEAM_INDEX] + HOME_GOAL;
            MATCH_PLAYED[VISITING_TEAM_INDEX] = MATCH_PLAYED[VISITING_TEAM_INDEX] + 1;

            // CHECK WINNER

            // When Home Team is winner
            if (HOME_GOAL > VISITING_GOAL) {
                set_for_win(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX, VISITING_TEAM_INDEX);
            }
            // 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) {
                set_for_lost(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX, VISITING_TEAM_INDEX);
            }
            // WHEN MATCH IS DRAW
```

```cpp
            else if (HOME_GOAL == VISITING_GOAL) {
                set_for_draw(WIN, LOST, DRAW, POINT, HOME_TEAM_INDEX, VISITING_TEAM_INDEX);
            }

            cout << "PLEASE PRESS ENTER TO RETURN TO MAIN MENU";
            cin.get();
        }
        else if (MENU_OPTION == "PRINT") {

            size_t large_print_space = 20;
            size_t medium_print_space = 13;
            size_t small_print_space = 11;

            int MAXIMUM_POINT = 0;
            int POINT_COPY[SIZE], POINT_SORTED[SIZE];

            size_t INDEX_SORTED[SIZE];

            array_initialize(POINT_COPY, SIZE);
            array_initialize(POINT_SORTED, SIZE);
            array_initialize(INDEX_SORTED, SIZE);

            //COPY THE POINT VALUE IN THE POINT_COPY SO THAT MAX FUNCTION COMPARES VALUE>0
            for (size_t counter0 = 0; counter0 < get_length(TEAM); counter0++) {
                int TEMPORARY = 1;
                POINT_COPY[counter0] = POINT[counter0] + TEMPORARY;
            }


            for (size_t counter1 = 0; counter1 < get_length(TEAM); counter1++) {
                INDEX_SORTED[counter1] = find_maxvalue(POINT_COPY, get_length(TEAM), MAXIMUM_POINT);
                POINT_SORTED[counter1] = POINT[INDEX_SORTED[counter1]];
                POINT_COPY[INDEX_SORTED[counter1]] = 0;
            }

            cout << endl << endl;
            print_table_coloumn_for_number(medium_print_space);
            print_table_banner();

            for (size_t TEAM_PRINT_INDEX = 0; TEAM_PRINT_INDEX < get_length(TEAM); TEAM_PRINT_INDEX++) {
                print_table_coloumn_for_number(medium_print_space);
                cout << TEAM[INDEX_SORTED[TEAM_PRINT_INDEX]];
                print_table_coloumn_for_name(get_length(TEAM[INDEX_SORTED[TEAM_PRINT_INDEX]]));
                cout << setw(5) << MATCH_PLAYED[INDEX_SORTED[TEAM_PRINT_INDEX]];
                print_table_coloumn_for_number(medium_print_space);
```

```
                    cout << setw(5) << FOR_GOAL[INDEX_SORTED[TEAM_PRINT_INDEX]] 
                      << "-" << AGAINST_GOAL[INDEX_SORTED[TEAM_PRINT_INDEX]];
                    print_table_coloumn_for_number(medium_print_space);
                    cout << setw(3) << WIN[INDEX_SORTED[TEAM_PRINT_INDEX]];
                    print_table_coloumn_for_number(small_print_space);
                    cout << LOST[INDEX_SORTED[TEAM_PRINT_INDEX]];
                    print_table_coloumn_for_number(small_print_space);
                    cout << DRAW[INDEX_SORTED[TEAM_PRINT_INDEX]];
                    print_table_coloumn_for_number(small_print_space);
                    cout << setw(2) << POINT[INDEX_SORTED[TEAM_PRINT_INDEX]] << 
                      endl;
                }

                cout << endl << endl;
                cout << "PLEASE PRESS ENTER TO RETURN TO MAIN MENU";
                cin.get();

            }
            else if (MENU_OPTION == "CLEAR") {

                cout << setw(80) << "TO CLEAN PLEASE ENTER NOW" << endl;
                cin.get();
                system("CLS");
                cout << setw(80) << "PLEASE ENTER AGAIN TO RETURN TO THE MAIN 
                  MENU" << endl;
                cin.get();
                system("CLS");
            }
            else if (MENU_OPTION != "EXIT") {
                cout << "INVALID INPUT, PLEASE TRY AGAIN THANKS" << endl;
                cin.clear();
            }
        } while (MENU_OPTION != "EXIT");

        cout << endl << endl;
        system("pause;");
        return 0;
    }
```

```
#include<iostream>
#include<string>


using namespace std;

//*******************************************************************
// FUNCTION DEFINATION FOR string_to_upper                          *
// THE FUNCTION TAKES A STRING AS AN ARGUMENT AND CONVERTS          *
// EACH CHARACHTER OF THE STRING INTO UPPER CHARACTER.              *
// THE FUNCTION RETURNS THE CONVERTED STRING.                       *
//*******************************************************************

string string_to_upper(string TEXT) {
    for (size_t index = 0; index < TEXT.length(); index++) {
        TEXT[index] = toupper(TEXT[index]); // CALLS THE toupper FUNCTION
          WITH CHAR INPUT AND RETURNED VALUE IS SAVED IN TEXT[index].
    }
    return TEXT;
}

```

```
#include<iostream>
#include<string>
#include<iomanip>

using namespace std;

//*******************************************************************
// FUNCTION DEFINATION FOR OVERLOAD FUNCTION array_initialize        *
// THIS FUNCTION TAKES INTEGER ARRAY AND THE SIZE OF THE ARRAY AS   *
// PARAMETER AND INITIALIZE WITH 0 VALUE UNTIL THE SIZE.            *
//*******************************************************************
void array_initialize(int ARRAY[], size_t LENGTH) {
    for (size_t ARRAYINDEX = 0; ARRAYINDEX < LENGTH; ARRAYINDEX++) {
        ARRAY[ARRAYINDEX] = 0;
    }
}

//************************************************************************
// FUNCTION DEFINATION FOR OVERLOAD FUNCTION array_initialize            *
// THIS FUNCTION TAKES size_t(unsigned int) ARRAY AND THE SIZE OF        *
// THE ARRAY AS FUNCTION PARAMETER AND INITIALIZE WITH 0 VALUE UNTIL     *
// THE SIZE.                                                             *
//************************************************************************
void array_initialize(size_t ARRAY[], size_t LENGTH) {
    for (size_t ARRAYINDEX = 0; ARRAYINDEX < LENGTH; ARRAYINDEX++) {
        ARRAY[ARRAYINDEX] = 0;
    }
}

//************************************************************************
// FUNCTION DEFINATION FOR add_team_in_table                             *
// THIS FUNCTION TAKES STRING ARRAY, STRING AND THE INDEX OF THE ARRAY   *
// WHERE THE TEAM NAME WILL BE SAVED.                                    *
//************************************************************************
void add_team_in_table(string TEAM_TABLE[], string TEAM_NAME, size_t INDEX) 
  {
    TEAM_TABLE[INDEX] = TEAM_NAME;
    cout << setw(80) << "THANKS! NEW TEAM IS ADDED TO THE TABLE" << endl << 
      endl;
}

//                                                                          
  ***************************************************************************
  *
// 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    
  *
// RETURNS THE INDEX OF MATCED TEAM. IF TEAM DOES NOT EXIST IN THE TABLE IT 
```

```cpp
*
// RETURNS 0.                                                              *
//***************************************************************************
size_t check_team_in_table(string TEAM_TABLE[], string TEAM_NAME, size_t LENGTH) {

    size_t index_to_return = 0;
    for (size_t counter0 = 0; counter0 < LENGTH; counter0++) {

        if (TEAM_NAME == TEAM_TABLE[counter0]) {
            cout << setw(80) << "ENTERED TEAM IS ALREADY IN THE TABLE" << endl << endl;
            index_to_return = counter0;
        }
    }
    //cout << index_to_return << endl;
    return index_to_return;
}

//***************************************************************************
// FUNCTION DEFINATION FOR set_for_win                                     *
// THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS   *
// INPUT ARGUMENT TO SET THE FOLOWWING TABLE WHEN THE TEAM HAS OWN A MATCH. *                              *
//***************************************************************************
void set_for_win(int WIN[], int LOST[], int DRAW[], int POINT[], size_t INDEX1, size_t INDEX2) {

    WIN[INDEX1] = WIN[INDEX1] + 1;
    LOST[INDEX1] = LOST[INDEX1] + 0;
    DRAW[INDEX1] = DRAW[INDEX1] + 0;
    POINT[INDEX1] = POINT[INDEX1] + 3;

    WIN[INDEX2] = WIN[INDEX2] + 0;
    LOST[INDEX2] = LOST[INDEX2] + 1;
    DRAW[INDEX2] = DRAW[INDEX2] + 0;
    POINT[INDEX2] = POINT[INDEX2] + 0;
}

//***************************************************************************
// FUNCTION DEFINATION FOR set_for_lost                                    *
```

```c
// THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS *
// INPUT ARGUMENT TO SET THE FOLOWWING TABLE WHEN THE TEAM HAS LOST A MATCH.*                                                           *
//***************************************************************************
void set_for_lost(int WIN[], int LOST[], int DRAW[], int POINT[], size_t INDEX1, size_t INDEX2) {

    WIN[INDEX1] = WIN[INDEX1] + 0;
    LOST[INDEX1] = LOST[INDEX1] + 1;
    DRAW[INDEX1] = DRAW[INDEX1] + 0;
    POINT[INDEX1] = POINT[INDEX1] + 0;

    WIN[INDEX2] = WIN[INDEX2] + 1;
    LOST[INDEX2] = LOST[INDEX2] + 0;
    DRAW[INDEX2] = DRAW[INDEX2] + 0;
    POINT[INDEX2] = POINT[INDEX2] + 3;
}

//******************************************************************************
// FUNCTION DEFINATION FOR set_for_draw                                            *
// THIS FUNCTION TAKES MULTIPLE INT ARRAY AND THE LENGTH OF THIS ARRAY AS          *
// INPUT ARGUMENT TO SET THE FOLLOWING TABLE WHEN THE TEAMs HAS DRAWN A MATCH.   *                                                           *
//******************************************************************************
void set_for_draw(int WIN[], int LOST[], int DRAW[], int POINT[], size_t INDEX1, size_t INDEX2) {

    WIN[INDEX1] = WIN[INDEX1] + 0;
    LOST[INDEX1] = LOST[INDEX1] + 0;
    DRAW[INDEX1] = DRAW[INDEX1] + 1;
    POINT[INDEX1] = POINT[INDEX1] + 1;

    WIN[INDEX2] = WIN[INDEX2] + 0;
    LOST[INDEX2] = LOST[INDEX2] + 0;
    DRAW[INDEX2] = DRAW[INDEX2] + 1;
    POINT[INDEX2] = POINT[INDEX2] + 1;
}

//******************************************************************************
// FUNCTION DEFINATION FOR find_maxvalue                                           *
// THIS FUNCTION TAKES INT ARRAY, LENGTH OF THE ARRAY AND MAXIMUM POINT AS
```

```
      *
// THE INPUT ARGUMENT AND RETURNS THE INDEX OF THE MAXIMUM VALUE OF THE
  ARRAY.  *                                                            *
//
  **************************************************************************
  *****
size_t find_maxvalue(int POINTCOPY[], size_t LENGTH, int MAXIMUM_POINT) {

    size_t index0 = 0;
    for (size_t counter1 = 0; counter1 < LENGTH; counter1++) {
        if (POINTCOPY[counter1] > MAXIMUM_POINT) {
            MAXIMUM_POINT = POINTCOPY[counter1];
            index0 = counter1;
        }
    }
    return index0;
}
```

```
#include <iostream>
#include<string>

using namespace std;

//******************************************************************
// FUNCTION DEFINATION FOR OVERLOAD FUNCTION get_length             *
// THIS FUNCTION USES AN STRING PARAMETER AND                      *
// RETURNS THE LENGTH AS SIZE_T                                    *
//******************************************************************

size_t get_length(string STRING) {

    size_t LENGTH = STRING.length();
    return LENGTH;
}


//******************************************************************
// FUNCTION DEFINATION FOR OVERLOAD FUNCTION get_length             *
// THIS FUNCTION USES AN STRING ARRAY AS PARAMETER AND             *
// RETURNS THE LENGTH AS SIZE_T                                    *
//******************************************************************

size_t get_length(string STRINGARRAY[]) {
    size_t LENGTH_ARRAY = 0;
    while (!STRINGARRAY[LENGTH_ARRAY].empty()) {
        ++LENGTH_ARRAY;
    }
    return LENGTH_ARRAY;
}


```

```cpp
#include<iostream>
#include<iomanip>


using namespace std;


// *****************************************************************************
// FUNCTION DEFINATION FOR print_menu_option                                   *
// THE FUNCTION PRINTS THE FOLLOWING AVAILABLE OPTIONS IN THE MENU             *
// ADD: TO ADD MATCH INFORMATION SUCH AS ADD TEAM RECORDS, UPDATE POINT TABLE  *
// PRINT: TO PRINT OUT THE CURRENT TEAM STANDING                               *
// CLEAR: TO CLEAR THE SCREEN                                                  *
// EXIT: TO EXIT FROM THE PROGRAM                                              *
// *****************************************************************************
void print_menu_option() {

    // Get the Menu option from the user
    cout << setw(85) << "PLEASE ENTER YOUR CHOICE FROM THE BELOW OPTIONS \n" << endl;
    cout << endl;
    cout << setw(100) << "__________________________________________________________________________\n";
    cout << setw(94) << "|OPTIONS    |                       DESCRIPTIONS                 |\n";
    cout << setw(101) << "|**************|*********************************************************| \n";
    cout << setw(70) << "|ADD        |   TO ADD SCORE OF THE NEW MATCH.              |\n";
    cout << setw(87) << "|PRINT      |   TO PRINT THE STANDING TABLE OF THE CHAMPIONSHIP.  |\n";
    cout << setw(63) << "|CLEAR      |   TO CLEAR THE SCREEN.                  |\n";
    cout << setw(66) << "|EXIT       |   TO EXIT FROM THE PROGRAM.             |\n";
    cout << setw(99) << "|______________|_________________________________________________________|";

    cout << endl << endl;

}

```

```
//******************************************************************************
// FUNCTION DEFINATION FOR print_add_banner                                    *
// THE FUNCTION PRINTS THE FOLLOWING MESSAGE IN THE PRINTING MENU              *
//******************************************************************************
void print_add_banner() {
    cout << setw(81) << "!!!WELCOME TO THE CHAMPIONSHIP!!!" << endl;
    cout << setw(80) << "PLEASE ENTER THE FOLLOWING INFROMATION" << endl;
}

//**********************************************************************************************
// FUNCTION DEFINATION FOR print_table_coloumn_for_name                                       *
// THE FUNCTION PRINTS WHITESPACE ACCORDING TO THE size_t SPACE VARIABLE SPECIFIED            *
//  AS FUNCTION INPUT TARGUMENT TO PROVIDE APROPRIATE SPACE DEPENDING ON THE NAME OF THE TEAM *
//**********************************************************************************************
void print_table_coloumn_for_name(size_t SPACE) {

    size_t possible_highest_team_name = 20;
    for (size_t counter = 0; counter < possible_highest_team_name - SPACE; counter++) {
        cout << " ";
    }
}

//*********************************************************************************************
// FUNCTION DEFINATION FOR print_table_coloumn_for_number                                    *
// THE FUNCTION PRINTS SPACE IN THE TEAM STANDING TABLE IN THE PRINT MENU.                   *
// THIS FUNCTION PRINTS WHITESPACE ACCORDING TO THE NUMBER SPECIFIEDBY size_t SPACE VARIABLE *
//*********************************************************************************************
void print_table_coloumn_for_number(size_t SPACE) {
    for (size_t counter = 0; counter < SPACE; counter++) {
        cout << " ";
    }
}
```

```cpp

//***************************************************************************************
// FUNCTION DEFINATION FOR print_table_coloumn_for_name             *
// THE FUNCTION PRINTS THE FOLLOWING BANNER FOR THE TEAM STANDING IN THE PRINT MENU     *
//***************************************************************************************
void print_table_banner() {

    size_t print_name = 8;
    size_t print_number = 12;
    cout << "TEAM";
    print_table_coloumn_for_name(print_name);
    cout << "Match Played";
    print_table_coloumn_for_name(print_number);
    cout << "FOR-AGAINST";
    print_table_coloumn_for_name(print_number);
    cout << "WIN";
    print_table_coloumn_for_name(print_number);
    cout << "LOST";
    print_table_coloumn_for_name(print_number);
    cout << "DRAW";
    print_table_coloumn_for_name(print_number);
    cout << "POINTS" << endl;
    print_table_coloumn_for_name(print_number);
    cout << "************************************************************************************************************\n" << endl;
}


```