Write a C/C++ program to the process the Gym data using the following constraints:
i. Store ID, Height and Weight of each member
ii. A member can be added/removed/updated
iii. The program should be menu operated
iv. Define a structure with data members ID, Height and Weight.
v. Calculate average Height of the members
vi. Calculate average Weight of the members
vii. Calculate Max Height and Weight
viii. Calculate Min Height and Weight
ix. Display BMI classification of a given member (use following table)

Source Code :

```
#include <bits/stdc++.h>
using namespace std;

class Member {
    int id;
    float height, weight;
    string pass;

public:
    void SetData(int i, float h, float w,
string p) { id = i; height = h; weight = w;
pass = p; }

    bool Auth() {
        string s;
        for (int i = 0; i < 3; ++i) {
            cout << "Enter Password: "; cin
>> s;
            if (s == pass) return true;
        }
        return false;
    }

    void Update() {
        if (!Auth()) return void(cout <<
"Authentication failed!\n\n");
        float h_ft;
        cout << "New Height (ft): "; while
(!(cin >> h_ft)) { cin.clear();
cin.ignore(1000, '\n'); cout << "Invalid
input. Try again: "; }
        height = h_ft * 0.3048;
        cout << "New Weight: "; while (!(cin
>> weight)) { cin.clear(); cin.ignore(1000,
'\n'); cout << "Invalid input. Try again: ";
}
        cout << "Updated successfully!\n\n";
    }

    void ShowBMI() {
        if (!Auth()) return void(cout <<
"Authentication failed!\n\n");
        float bmi = weight / (height *
height);
        cout << "BMI = " << bmi <<
"\nClassification: ";
        if (bmi < 18.5) cout <<
"Underweight\n\n";
        else if (bmi < 25) cout <<
"Normal\n\n";
        else if (bmi < 30) cout <<
"Overweight\n\n";
        else cout << "Obese\n\n";
    }
```

```
    float getHeight() const { return height;
}
    float getWeight() const { return weight;
}
};

Member members[1000];
int total = 1;

void Pause() { string s; cout << "<---Press
any key--->\n"; cin >> s; system("cls"); }

float InputFloat(const string &msg) {
    float val;
    cout << msg;
    while (!(cin >> val)) {
        cin.clear(); cin.ignore(1000, '\n');
        cout << "Invalid input. Try again:
";
    }
    return val;
}

void AddMember() {
    float h = InputFloat("Height (ft): ") *
0.3048;
    float w = InputFloat("Weight (kg): ");
    string pass;
    cout << "Set Password: "; cin >> pass;
    members[total].SetData(total, h, w,
pass);
    cout << "Member ID: " << total + 1000 <<
"\nAdded Successfully\n\n";
    total++; Pause();
}

void UpdateMember() {
    int id; cout << "Member ID: ";
    if (!(cin >> id)) { cin.clear();
cin.ignore(1000, '\n'); cout << "Invalid
input!\n\n"; return Pause(); }
    int idx = id - 1000;
    if (idx > 0 && idx < total)
members[idx].Update();
    else cout << "Member not found!\n\n";
    Pause();
}

void RemoveMember() {
    int id; cout << "Member ID: ";
    if (!(cin >> id)) { cin.clear();
cin.ignore(1000, '\n'); cout << "Invalid
input!\n\n"; return Pause(); }
    int idx = id - 1000;
    if (idx > 0 && idx < total &&
members[idx].Auth()) {
        for (int i = idx; i < total - 1;
i++) members[i] = members[i + 1];
        total--; cout << "Member Removed
Successfully!\n\n";
    } else cout << "Authentication failed or
Member not found.\n\n";
    Pause();
}

void StatHW(bool max) {
    if (total == 1) return void(cout << "No
members yet.\n\n", Pause());
    float h = max ? 0 : 1e9, w = h;
    for (int i = 1; i < total; i++) {
        h = max ? std::max(h,
members[i].getHeight()) : std::min(h,
members[i].getHeight());
        w = max ? std::max(w,
members[i].getWeight()) : std::min(w,
members[i].getWeight());
    }
    cout << (max ? "Max" : "Min") << "
Height: " << h << " m\n" << (max ? "Max" :
"Min") << " Weight: " << w << " kg\n\n";
    Pause();
}

void AvgHW() {
    if (total == 1) return void(cout << "No
members yet.\n\n", Pause());
    float th = 0, tw = 0;
    for (int i = 1; i < total; i++) th +=
members[i].getHeight(), tw +=
members[i].getWeight();
    cout << "Average Height: " << th /
(total - 1) << " m\n";
    cout << "Average Weight: " << tw /
(total - 1) << " kg\n\n";
    Pause();
}

void BMI() {
    int id; cout << "Member ID: ";
    if (!(cin >> id)) { cin.clear();
cin.ignore(1000, '\n'); cout << "Invalid
input!\n\n"; return Pause(); }
    int idx = id - 1000;
    if (idx > 0 && idx < total)
members[idx].ShowBMI();
    else cout << "Member not found!\n\n";
    Pause();
}

int main() {
    while (true) {
        cout << "<---Main Menu--->\n\n";
        cout << "1. Add Member\n2. Update
Member\n3. Remove Member\n4. Max Height &
Weight\n";
        cout << "5. Min Height & Weight\n6.
Average Height & Weight\n7. BMI
Classification\n0. Exit\n\n";
        cout << "Choose an option: ";
        int choice;
        if (!(cin >> choice)) { cin.clear();
cin.ignore(1000, '\n'); cout << "Invalid
input!\n\n"; continue; }
        system("cls");
        switch (choice) {
            case 1: AddMember(); break;
            case 2: UpdateMember(); break;
            case 3: RemoveMember(); break;
            case 4: StatHW(true); break;
            case 5: StatHW(false); break;
            case 6: AvgHW(); break;
            case 7: BMI(); break;
            case 0: return 0;
            default: cout << "Invalid
option!\n\n";
        }
    }
}
```

Input :

```
Choose an option: 1
Height (ft): 5.9
Weight (kg): 70
Set Password: abc123
<---Press any key---> xyz
Choose an option: 7
Member ID: 1001
Enter Password: abc123
<---Press any key---> done
```

Output :

```
<---Main Menu--->
1. Add Member
2. Update Member
3. Remove Member
4. Max Height & Weight
5. Min Height & Weight
6. Average Height & Weight
7. BMI Classification
0. Exit
Choose an option: 1
Height (ft): 5.9
Weight (kg): 70
Set Password: abc123
Member ID: 1001
Added Successfully
<---Press any key--->
<---Main Menu--->

1. Add Member
2. Update Member
3. Remove Member
4. Max Height & Weight
5. Min Height & Weight
6. Average Height & Weight
7. BMI Classification
0. Exit

Choose an option: 7
Member ID: 1001
Enter Password: abc123
BMI = 24.45
Classification: Normal

<---Press any key--->
```