Problem 1:

Write a Circle class. Now do the following
i) Initialize radius of 3 circles
ii) Find area of all of them
iii) Find the total area

Source Code :

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

class Circle{
    int radius;
    float area;
    double sum = 0;

    public :
        void GetInput(){
            int a;
            cin >> a;
            radius = a;
        }

        void FindArea(){
            float x = 3.1416 * radius * radius;
            area = x;
        }

        void PrintValue(){
            cout << "Radius : " << radius << '\n';
            cout << "Area : " << area << '\n';
        }

        void CalSum(){
            sum += area;
        }

        void PrintSum(){
            cout << '\n' << "Total Sum : " << sum << '\n';
        }
};

int main(void){
    Circle c;

    for (int i = 0; i < 3; i++){
        c.GetInput();
        c.FindArea();
        c.PrintValue();
    }
```

```
    c.CalSum();
    c.PrintSum();

}
```

Input :

```
5 10 7
```

Output :

```
Radius : 5
Area : 78.54
Radius : 10
Area : 314.16
Radius : 7
Area : 153.938

Total Sum : 153.938
```

Problem 02:

Write a Triangle class. Now do the following
i) Initialize edges of a triangle
ii) Find area of the triangle
iii) Check whether the 3 edges form a triangle

Source Code :

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

class Triangle {
    int edge1, edge2, edge3;
    float area = 0;
    bool isOk = false;

public:
    void GetInput() {
        cin >> edge1 >> edge2 >> edge3;
    }

    void CanForm() {
        if ((edge1 + edge2 > edge3) && (edge2 + edge3 > edge1) && (edge1 + edge3 >
edge2))
            isOk = true;
        else
            isOk = false;
        PrintValue("decision");
    }
```

```
    void FindArea() {
        if (isOk) {
            float s = (edge1 + edge2 + edge3) / 2.0;
            area = sqrt(s * (s - edge1) * (s - edge2) * (s - edge3));
        } else {
            area = 0;
        }
    }

    void PrintValue(string s) {
        if (s == "area") {
            if (isOk)
                cout << "Area : " << area << '\n';
        } else if (s == "decision") {
            if (isOk)
                cout << "It can form triangle.\n";
            else
                cout << "It can't form triangle.\n";
        }
    }
};

int main() {
    Triangle t;
    t.GetInput();
    t.CanForm();
    t.FindArea();
    t.PrintValue("area");
    return 0;
}
```

Input :

```
3 4 5
```

Output :

```
It can form triangle.
Area : 6
```

Problem 03:

Write a Account class. Now do the following
i) Initialize 5 accounts
ii) Deposit money to an account
iii) Withdrawal money from an account
iv) Transfer money from one account to another

Source Code :

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

```
class Bank {
    int AccNo;
    float Balance;

public:
    void SetData(int n, float b) {
        AccNo = n;
        Balance = b;
    }

    float GetBalance() {
        return Balance;
    }

    void SetBalance(float b) {
        Balance += b;
    }

    void SetWithdrawal(float b) {
        if (b <= Balance)
            Balance -= b;
        else
            cout << "Insufficient balance!\n";
    }

    int GetAccNo() {
        return AccNo;
    }
};

Bank b[5];
int Total = 0;

void Create() {
    if (Total >= 5) {
        cout << "Max 5 accounts allowed.\n";
        return;
    }

    int acc;
    float bal;
    cout << "Enter Account Number: ";
    cin >> acc;
    cout << "Enter Initial Balance: ";
    cin >> bal;
    b[Total++].SetData(acc, bal);
    cout << "Account created successfully.\n";
}

int FindIndex(int acc) {
    for (int i = 0; i < Total; ++i) {
        if (b[i].GetAccNo() == acc)
            return i;
    }
```

```
    return -1;
}

void Deposit() {
    int acc;
    float amount;
    cout << "Enter Account Number: ";
    cin >> acc;
    int index = FindIndex(acc);
    if (index == -1) {
        cout << "Account not found.\n";
        return;
    }
    cout << "Enter Deposit Amount: ";
    cin >> amount;
    b[index].SetBalance(amount);
    cout << "Deposit successful.\n";
}

void Withdrawal() {
    int acc;
    float amount;
    cout << "Enter Account Number: ";
    cin >> acc;
    int index = FindIndex(acc);
    if (index == -1) {
        cout << "Account not found.\n";
        return;
    }
    cout << "Enter Withdrawal Amount: ";
    cin >> amount;
    b[index].SetWithdrawal(amount);
}

void Transfer() {
    int fromAcc, toAcc;
    float amount;
    cout << "Enter From Account Number: ";
    cin >> fromAcc;
    cout << "Enter To Account Number: ";
    cin >> toAcc;

    int fromIndex = FindIndex(fromAcc);
    int toIndex = FindIndex(toAcc);

    if (fromIndex == -1 || toIndex == -1) {
        cout << "Invalid account number(s).\n";
        return;
    }

    cout << "Enter Amount to Transfer: ";
    cin >> amount;

    if (b[fromIndex].GetBalance() >= amount) {
        b[fromIndex].SetWithdrawal(amount);
```

```
        b[toIndex].SetBalance(amount);
        cout << "Transfer successful.\n";
    } else {
        cout << "Insufficient balance.\n";
    }
}

void ShowBalance() {
    int acc;
    cout << "Enter Account Number: ";
    cin >> acc;
    int index = FindIndex(acc);
    if (index == -1) {
        cout << "Account not found.\n";
        return;
    }
    cout << "Current Balance: " << b[index].GetBalance() << '\n';
}

int main() {
    int option;
    while (true) {
        cout << "\n<----- MAIN MENU ----->\n";
        cout << "1. New Account\n";
        cout << "2. Deposit\n";
        cout << "3. Withdrawal\n";
        cout << "4. Transfer Money\n";
        cout << "5. Show Balance\n";
        cout << "6. Exit\n";
        cout << "Enter Your Option: ";
        cin >> option;

        if (option == 1) Create();
        else if (option == 2) Deposit();
        else if (option == 3) Withdrawal();
        else if (option == 4) Transfer();
        else if (option == 5) ShowBalance();
        else if (option == 6) {
            cout << "Program exits.\n";
            break;
        }
        else cout << "Invalid option.\n";
    }

    return 0;
}
```

Input & Output :

```
<----- MAIN MENU ----->
1. New Account
2. Deposit
3. Withdrawal
4. Transfer Money
```

```
5. Show Balance
6. Exit
Enter Your Option: 2

Enter Account Number: 101
Enter Deposit Amount: 1500
Deposit successful.

<----- MAIN MENU ----->
1. New Account
2. Deposit
3. Withdrawal
4. Transfer Money
5. Show Balance
6. Exit
Enter Your Option: 5

Enter Account Number: 101
Current Balance: 6500

<----- MAIN MENU ----->
1. New Account
2. Deposit
3. Withdrawal
4. Transfer Money
5. Show Balance
6. Exit
Enter Your Option: 6

Program exits.
```

Problem - 04

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. DeEine 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 :

```cpp
#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 = " << fixed << setprecision(2) << 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 ? max(h, members[i].getHeight()) : min(h, members[i].getHeight());
        w = max ? max(w, members[i].getWeight()) : 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 (2) {
        cout << "<---Main Menu--->\n\n";
        cout << "1. Add Member\n";
        cout << "2. Update Member\n";
        cout << "3. Remove Member\n";
        cout << "4. Max Height & Weight\n";
        cout << "5. Min Height & Weight\n";
        cout << "6. Average Height & Weight\n";
        cout << "7. BMI Classification\n";
        cout << "0. 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 & 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
8. Exit

Choose an option: 1
Height (ft): 5.9
Weight (kg): 70
Set Password: abc123
Member ID: 1001
Added Successfully

<---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
8. Exit
```

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

<---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
8. Exit

Choose an option: 0

Program exited.
```