Module 1:

Problem Statement: Want to feel better, have more energy and even add years to your life? Just exercise. Joining a gym can help you stay motivated to exercise consistently. This is a great way to build muscle, lose weight, lower blood pressure, boost mental focus, and more. Over time, you can look better, feel better, and accomplish things you never thought possible! It is important to know your BMI (Body Mass Index) for good health. (for more read below) 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)

Code:

```
#include <iostream>
#include <conio.h>
using namespace std;
struct Member
{
    int id;
    float height;
    float weight;
    bool exist;
};
int total = 0;
struct Member mx[10000];
int menu()
{
    int op;
    cout << "******Main Menu*****" << endl;
    cout << "1. Add Member" << endl;
    cout << "2. Update Member" << endl;
    cout << "3. Remove Member" << endl;
    cout << "4. Max Height and Weight" << endl;
    cout << "5. Min Height and Weight" << endl;
    cout << "6. Average Height and Weight" << endl;
    cout << "7. BMI Classification" << endl;
    cout << "8. Display all" << endl;
    cout << "9. Exit" << endl;
```

```
    cout << "   Enter Your Option(1-8): ";
    cin >> op;
    return (op);
}
int serial(int y)
{
    int index = -1;
    for (int i = 0; i < total; i++)
    {
        if (y == mx[i].id)
        {
            index = i;
        }
    }
    return index;
}
int searchMember(int y)
{
    int i;
    int flag = -1;
    for (i = 0; i < total; i++)
    {
        if (mx[i].id == y)
        {
            cout << "found" << endl;
            flag = 1;
        }
    }
    return flag;
}
int searchMember2(int y)
{
    int i;
    int flag = -1;
    for (i = 0; i <= total; i++)
    {
        if (mx[i].id == y)
        {
            cout << "found" << endl;
            flag = 1;
        }
    }
    return flag;
}
void addMember()
```

```
{
    total++;
    cout << "Enter ID: ";
    cin >> mx[total].id;
    int check = searchMember(mx[total].id);
    if (check != 1)
    {
        cout << "Enter Height: ";
        cin >> mx[total].height;
        cout << "Enter Weight: ";
        cin >> mx[total].weight;
        mx[total].exist = true;
    }
    else
    {
        total--;
        cout << "Already exist" << endl;
    }
    getch();
}
void updateMember()
{
    int x, idx;
    cout << "Enter Member ID: " << endl;
    cin >> x;
    total++;
    idx = searchMember(x);
    int serialno = serial(x);
    if (idx != 1)
    {
        cout << "Sorry Member NOT found..." << endl;
    }
    else
    {
        cout << "Enter new height: " << endl;
        cin >> mx[serialno].height;
        cout << "Enter new weight: " << endl;
        cin >> mx[serialno].weight;
        total--;
    }
    getch();
}

void removeMember()
{
```

```
    int x, idx;
    cout << "Enter Member ID: " << endl;
    cin >> x;
    total++;
    idx = searchMember(x);
    total--;
    int serialno = serial(x);
    if (idx != 1)
    {
        cout << "Sorry Member NOT found..." << endl;
    }
    else
    {
        for (int i = serialno; i <= total; i++)
        {
            mx[serialno].id = mx[serialno + 1].id;
            mx[serialno].height = mx[serialno + 1].height;
            mx[serialno].weight = mx[serialno + 1].weight;
        }
        total--;
    }
    getch();
}
void maxHeightWeight()
{
    float maxHeight = mx[1].height;
    float maxWeight = mx[1].weight;
    int maxHeightId = mx[1].id, maxWeightId = mx[1].id;
    for (int i = 2; i <= total; i++)
    {
        if (mx[i].height > maxHeight)
        {
            maxHeight = mx[i].height;
            maxHeightId = mx[i].id;
        }
        if (mx[i].weight > maxWeight)
        {
            maxWeight = mx[i].weight;
            maxWeightId = mx[i].id;
        }
    }
    cout << "Maximum Height: " << maxHeight << " is of member id: " << maxHeightId
<< endl;
    cout << "Maximum Weight: " << maxWeight << " is of member id: " << maxWeightId
<< endl;
```

```
    getch();
}
void minHeightWeight()
{
    float minHeight = mx[1].height;
    float minWeight = mx[1].weight;
    int minHeightId = mx[1].id, minWeightId = mx[1].id;
    for (int i = 2; i <= total; i++)
    {
        if (mx[i].height < minHeight)
        {
            minHeight = mx[i].height;
            minHeightId = mx[i].id;
        }
        if (mx[i].weight < minWeight)
        {
            minWeight = mx[i].weight;
            minWeightId = mx[i].id;
        }
    }
    cout << "Minimum height: " << minHeight << " is of member index " << minHeightId
<< endl;
    cout << "Minimum weight: " << minWeight << " is of member index " << minWeightId
<< endl;

    getch();
}
void avgHeightWeight()
{
    float totalHeight = 0, totalWeight = 0;
    for (int i = 1; i <= total; i++)
    {
        totalHeight += mx[i].height;
        totalWeight += mx[i].weight;
    }
    float avgHeight = totalHeight / (float)total;
    float avgWeight = totalWeight / (float)total;
    cout << "Avg Height: " << avgHeight << endl;
    cout << "Avg Weight: " << avgWeight << endl;
    getch();
}
void bmi()
{
    int x, idx;
```

```
    cout << "Enter Member ID: " << endl;
    cin >> x;
    total++;
    idx = searchMember(x);
    int serialno = serial(x);
    total--;
    if (idx != 1)
    {
        cout << "Sorry Member NOT found..." << endl;
    }
    else
    {
        float target_height = mx[serialno].height;
        float target_weight = mx[serialno].weight;
        cout << (mx[serialno].height) << endl;
        float bmi = (target_weight) / (target_height * target_height);
        cout << "BMI: " << bmi << endl;
        cout << "BMI Classification: ";
        if (bmi < 16)
            cout << "Severe Thinness" << endl;
        if (bmi >= 16 && bmi <= 17)
            cout << "Moderate Thinness" << endl;
        if (bmi >= 17 && bmi <= 18.5)
            cout << "Mild Thinness" << endl;
        if (bmi >= 18.5 && bmi <= 25)
            cout << "Normal" << endl;
        if (bmi >= 25 && bmi <= 30)
            cout << "Overweight" << endl;
        if (bmi >= 30 && bmi <= 35)
            cout << "Obese Class I" << endl;
        if (bmi >= 35 && bmi <= 40)
            cout << "Obese Class II" << endl;
        if (bmi > 40)
            cout << "Obese Class III" << endl;
    }
    getch();
}
void displayAll()
{
    cout << "SL.\tID\tHEI.\tWEI." << endl;
    for (int i = 0; i <= total; i++)
    {
        if (mx[i].exist == true)
        {
            cout << i << "\t" << mx[i].id << "\t" << mx[i].height << "\t" <<
```

```
mx[i].weight << endl;
        }
    }
    getch();
}
int main()
{
    int option;

    while (true)
    {
        option = menu();
        switch (option)
        {
        case 1:
            addMember();
            break;
        case 2:
            updateMember();
            break;
        case 3:
            removeMember();
            break;
        case 4:
            maxHeightWeight();
            break;
        case 5:
            minHeightWeight();
            break;
        case 6:
            avgHeightWeight();
            break;
        case 7:
            bmi();
            break;
        case 8:
            displayAll();
            break;
        case 9:
            cout << "End of program Run....";
            exit(0);
            break;
            getch();
        default:
            cout << "Not available";
```

```
            exit(0);
            break;
        }
    }
}
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 1
Enter ID: 121
Enter Height: 1.52
Enter Weight: 60
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 1
Enter ID: 131
Enter Height: 1.78
Enter Weight: 80
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 1
Enter ID: 141
Enter Height: 1.63
Enter Weight: 70
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 4
Maximum Height: 1.78 is of member id: 131
Maximum Weight: 80 is of member id: 131
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 5
Minimum height: 1.52 is of member index 121
Minimum weight: 60 is of member index 121
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 8
SL.     ID      HEI.    WEI.
1       121     1.52    60
2       131     1.78    80
3       141     1.63    70
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 6
Avg Height: 1.64333
Avg Weight: 70
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 2
Enter Member ID:
121
found
Enter new height:
1.70
Enter new weight:
65
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 8
SL.     ID      HEI.    WEI.
1       121     1.7     65
2       131     1.78    80
3       141     1.63    70
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 7
Enter Member ID:
131
found
found
1.78
BMI: 25.2493
BMI Classification: Overweight
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 3
Enter Member ID:
131
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 8
SL.     ID      HEI.    WEI.
1       121     1.52    60
2       141     1.63    70
```

```
******Main Menu*****
1. Add Member
2. Update Member
3. Remove Member
4. Max Height and Weight
5. Min Height and Weight
6. Average Height and Weight
7. BMI Classification
8. Display all
9. Exit
   Enter Your Option(1-8): 9
End of program Run....
```

Module 2 [Constructor, const & static]:

Problem Statement: The following class is a simple class for the operations in a Bank. The objects of Bank class stores id, amount of each client and n keeps track of total clients. Now extend the program do the following:

i) Write empty constructor to initialize id and amount to 0
ii) Write parameterized constructor to initialize id and amount
iii) Write copy constructor to Initialize id and amount
iv) Write a method to set id and amount
v) Write a method to change the amount only
vi) Write a method to display n only
vii) Write a method to display id, amount and n where their values can't be changed
viii) Create five clients with id and amount (use all constructors to do it)
ix) Find total amount of the bank

Code:

```
#include <iostream>
using namespace std;
class Bank
{
private:
    int id;
    float amount;
    static int n;
public:
    Bank()
    {
        id = 0;
        amount = 0;
        n++;
    }
    Bank(int x, float a)
    {
        id = x;
        amount = a;
        n++;
    }
    Bank(Bank &p)
    {
        id = p.id;
        amount = p.amount;
        n++;
    }
    void setData(int x, float a)
    {
```

```
        id = x;
        amount = a;
    }
    void changeData(float a)
    {
        amount = amount + a;
    }
    int getN()
    {
        return n;
    }
    float getAmount()
    {
        return (amount);
    }
    void display() const
    {
        cout << "Id: " << id << endl;
        cout << "Amount: " << amount << endl;
        cout << "Total Clients: " << n << endl;
    }
};
int Bank::n = 0;
int main()
{
    Bank b1(1, 400);
    b1.display();
    Bank b2(2, 200);
    b2.display();
    Bank b3;
    b3.setData(3, 500);
    b3.display();
    b3.changeData(100);
    b3.display();
    Bank b4(4, 50);
    Bank b5(5, 3000);
    float sum;
    sum = b1.getAmount();
    sum += b2.getAmount();
    sum += b3.getAmount();
    sum += b4.getAmount();
    sum += b5.getAmount();
    cout << "sum = " << sum << endl;
    cout << "Total clients: " << b3.getN() << endl;
}
```

```
Id: 1
Amount: 400
Total Clients: 1
Id: 2
Amount: 200
Total Clients: 2
Id: 3
Amount: 500
Total Clients: 3
Id: 3
Amount: 600
Total Clients: 3
sum = 4250
Total clients: 5
```