Assignment 02

2.1 Define a class to represent a bank account. Include the following members

Data members:

1. Name of the depositor

2. Account number

3. Type of account (savings or current)

4. Balance amount in the account

Member functions:

1. To assign initial values

2. To deposit an amount

3. To withdraw an amount after checking the balance

4. To display name and balance

Write a main function to test the program.

```
#include <iostream>
#include <string>
using namespace std;

class BankAccount {
private:
    string name;
    int accountNumber;
    string accountType;
    double balance;

public:
    void initialize(string depositorName, int accountNo, string type, double
initialBalance) {
        name = depositorName;
        accountNumber = accountNo;
        accountType = type;
        balance = initialBalance;
    }
```

```
    void deposit(double amount) {
        balance += amount;
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
        } else {
            cout << "Insufficient balance!" << endl;
        }
    }

    void display() {
        cout << "Account Holder: " << name << endl;
        cout << "Account Number: " << accountNumber << endl;
        cout << "Account Type: " << accountType << endl;
        cout << "Balance: " << balance << endl;
    }
};

int main() {
    BankAccount account;
    account.initialize("John Doe", 12345, "Savings", 1000.0);
    account.deposit(500.0);
    account.withdraw(300.0);
    account.display();
    return 0;
}
```

2.2 Modify the class and program of Problem No. 2.1 for handling of 10 customers.

```
#include <iostream>
#include <string>
using namespace std;

class BankAccount
{
private:
    string name;
```

```
    int accountNumber;
    string accountType;
    double balance;

public:
    void initialize(string depositorName, int accountNo, string type, double
initialBalance)
    {
        name = depositorName;
        accountNumber = accountNo;
        accountType = type;
        balance = initialBalance;
    }

    void deposit(double amount)
    {
        balance += amount;
    }

    void withdraw(double amount)
    {
        if (balance >= amount)
        {
            balance -= amount;
        }
        else
        {
            cout << "Insufficient balance!" << endl;
        }
    }

    void display()
    {
        cout << "Account Holder: " << name << endl;
        cout << "Balance: " << balance << endl;
    }
};

int main()
{
```

```
    BankAccount customers[10];

    for (int i = 0; i < 10; i++)
    {
        customers[i].initialize("Customer" + to_string(i + 1), 1000 + i, "Savings",
500.0);
    }

    customers[0].deposit(300.0);
    customers[1].withdraw(200.0);

    for (int i = 0; i < 10; i++)
    {
        cout << "Customer " << i + 1 << " details:" << endl;
        customers[i].display();
        cout << "-------------------------" << endl;
    }

    return 0;
}
```

2.3 Create two classes DM and DB which store the value of distances. DM stores distances in metres and centimetres and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB. Use a friend function to carry out the addition operation. The object that stores the results may be a DM object or DB object, depending on the units in which the results are required. The display should be in the format of feet and inches or metres and centimetres depending on the object on display

```
#include <iostream>
using namespace std;

class DB;

class DM
{
private:
    int meters;
```

```
    int centimeters;

public:
    void input(int m, int cm)
    {
        meters = m;
        centimeters = cm;
    }

    void display()
    {
        cout << "Distance in DM (Meters and Centimeters): " << meters << " meters "
<< centimeters << " cm" << endl;
    }

    friend void addDistance(DM &dmObj, DB &dbObj);
};

class DB
{
private:
    int feet;
    int inches;

public:
    void input(int ft, int in)
    {
        feet = ft;
        inches = in;
    }

    void display()
    {
        cout << "Distance in DB (Feet and Inches): " << feet << " feet " << inches
<< " inches" << endl;
    }

    friend void addDistance(DM &dmObj, DB &dbObj);
};
```

```
void addDistance(DM &dmObj, DB &dbObj)
{
    int totalCentimeters = dmObj.meters * 100 + dmObj.centimeters;
    int totalInches = dbObj.feet * 12 * 2.54 + dbObj.inches * 2.54;

    int totalCentimetersResult = totalCentimeters + totalInches;
    int resultMeters = totalCentimetersResult / 100;
    int resultCentimeters = totalCentimetersResult % 100;

    dmObj.meters = resultMeters;
    dmObj.centimeters = resultCentimeters;

    cout << "Result of addition (DM object): " << dmObj.meters << " meters " <<
dmObj.centimeters << " cm" << endl;
}

int main()
{
    DM dm1;
    DB db1;

    dm1.input(5, 60);
    db1.input(16, 4);

    dm1.display();
    db1.display();

    addDistance(dm1, db1);

    return 0;
}
```

2.4 Create a class distance with private data members km,m,cm. Include the following member

Functions:

input_distance() with 3 parameters to initialize the distance object.

display() to display a distance object.

add_distance() to perform addition of 2 distance objects and will return the result.

In the main function, create an array of n distance objects, initialize and display them.

Perform addition of any 2 distance object by prompting the user for the index of 2 objects in the array.

Display the result.

```
#include <iostream>
using namespace std;

class Distance {
    int km, m, cm;
public:
    void input(int k, int x, int c) {
        km=k; m=x; cm=c;
    }
    void display() {
        cout<<km<<" km "<<m<<" m "<<cm<<" cm\n";
    }
    Distance addDistance(Distance d) {
        int t=(km*100000+m*100+cm)+(d.km*100000+d.m*100+d.cm);
        Distance r; r.km=t/100000; r.m=(t%100000)/100; r.cm=t%100; return r;
    }
};

int main() {
    int n; cout<<"Enter number of distances: "; cin>>n;
    Distance *d=new Distance[n];

    for(int i=0;i<n;i++) {
        int k,x,c; cout<<"Distance "<<i+1<<" (km m cm): "; cin>>k>>x>>c;
        d[i].input(k,x,c);
    }

    cout<<"Distances:\n"; for(int i=0;i<n;i++) d[i].display();

    int a,b; cout<<"Enter indices to add: "; cin>>a>>b;
    d[a].addDistance(d[b]).display();
```

```
    delete[] d;
}
```