Principles of Programming Language (CS302)

Assignment - 1

U19CS012

1.) Create two classes DM and DB which store the value of distances.

- ✓ DM stores distances in metres and centimeters 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 centimeters depending on the object on display.

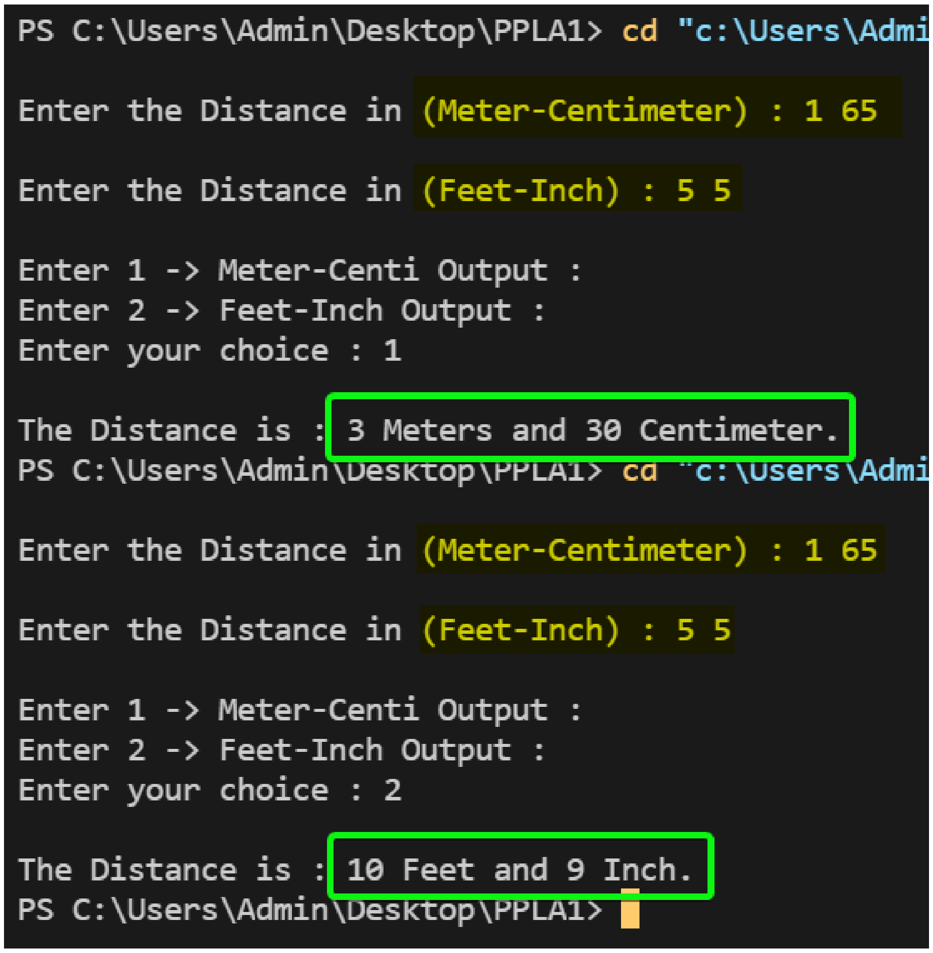
Code

```cpp
#include <iostream>
using namespace std;
// [U19CS012 - BHAGYA VINOD RANA]

// DB Class {Stores the Distance in Feet and Inches}
class DB;
// DM Class {Stores the Distance in Metres and Centimeters}
class DM;

// DM Class {Stores the Distance in Metres and Centimeters}
class DM
{
    double meter, centi;

public:
    void getdata()
    {
        cout << "\nEnter the Distance in (Meter-Centimeter) : ";
        cin >> meter >> centi;
    }
    void display()
    {
        cout << "\nThe Distance is : ";
        cout << meter << " Meters and " << centi << " Centimeter.";
    }
    friend void add(DM &, DB &);
```

```cpp
};

// DB Class {Stores the Distance in Feet and Inches}
class DB
{
    double inch, feet;

public:
    void getdata()
    {
        cout << "\nEnter the Distance in (Feet-Inch) : ";
        cin >> feet >> inch;
    }
    void display()
    {
        cout << "\nThe Distance is : ";
        cout << feet << " Feet and " << inch << " Inch.";
    }
    friend void add(DM &, DB &);
};

// Friend Functoin to Carry Out Addition Operation
void add(DM &a, DB &b);

int main()
{
    DM a;
    DB b;
    // Read Values from Class Objects
    a.getdata();
    b.getdata();
    // Call the Friend Function to Add Both the Objects in Different Units
    add(a, b);
}

// Friend Functoin to Carry Out Addition Operation
void add(DM &a, DB &b)
{
    int ch;
    cout << "\nEnter 1 -> Meter-Centi Output : ";
    cout << "\nEnter 2 -> Feet-Inch Output : ";
    cout << "\nEnter your choice : ";
    cin >> ch;

    if (ch == 1)
    {
        DM d;

        // Convert all to Common 'cm' Denominator
        // 1 Meter = 100 cm, 1 cm = 1 cm, 1 Feet = 30.48 cm, & Round Off to Nearest cm
        int c = ((a.meter * 100) + (a.centi) + (b.feet * 30.48) + (b.inch * 2.54));

        if (c >= 100)
        {
            d.meter = c / 100;
            d.centi = c % 100;
        }
        else
        {
            d.meter = 0;
            d.centi = c;
        }
        d.display();
    }
    else
    {
        DB d;
        // Convert all to Common 'inches' Denominator
        // 1 Meter = 39.3701 inch, 1 cm = 0.3937 inch, 1 Feet = 12 inch, & Round Off to
Nearest inch
        int i = ((a.meter * 39.3701) + (a.centi * 0.393701) + (b.feet * 12) + (b.inch));
        if (i >= 12)
        {
            d.feet = i / 12;
            d.inch = i % 12;
        }
        else
        {
            d.feet = 0;
            d.inch = i;
        }
        d.display();
    }
}
```

Output

Input	meter-centi	feet-inch	centimeter
Object1	1 m, 65 cm	5 feet, 5 inch	165 cm
Object2	1 m, 65 cm	5 feet, 5 inch	165 cm
Total	3 m, 30 cm	10 feet, 9 inch	330 cm



```
PS C:\Users\Admin\Desktop\PPLA1> cd "c:\Users\Admi

Enter the Distance in (Meter-Centimeter) : 1 65

Enter the Distance in (Feet-Inch) : 5 5

Enter 1 -> Meter-Centi Output :
Enter 2 -> Feet-Inch Output :
Enter your choice : 1

The Distance is : 3 Meters and 30 Centimeter.
PS C:\Users\Admin\Desktop\PPLA1> cd "c:\Users\Admi

Enter the Distance in (Meter-Centimeter) : 1 65

Enter the Distance in (Feet-Inch) : 5 5

Enter 1 -> Meter-Centi Output :
Enter 2 -> Feet-Inch Output :
Enter your choice : 2

The Distance is : 10 Feet and 9 Inch.
PS C:\Users\Admin\Desktop\PPLA1>
```

2.) Find errors, if any, in the following C++ statements.

a) long float x;

Error – **Yes**, <u>Too Many Datatypes</u>

Correction – long x; or float x;

b) char *cp = vp; // vp is a void pointer

Error – **Yes**, Pointer Type must be same on both side

Correction - char *cp = (char*) vp;

c) int code = three; // three is an enumerator

Error - **No**

d) int sp = new; // allocate memory with new

Error – **Yes**, syntax Error

Correction – int *p=new int[10];

e) enum (green, yellow, red);

Error – **Yes**, tag name missing.

Correction – enum **color**(green,yellow,red);

f) int const sp = total;

Error – **Yes**, address have to assign instead of content

Correction - int const* p = &total;

g) const int array_size;

Error – **Yes**, C++ requires a const to be initialized at time of defination

Correction - const int array_size = 5;

h) for (i=1; int i<10; i++) cout << i << "/n";

Error – **Yes,** undefined symbol i

Correction - for (int i=1; int i<10; i++) cout << i << "/n";

i) int &number = 100;

Error – **Yes,** invalid variable name

Correction - int number = 100;

j) float *p = new int 1101;

Error – **Yes**, wrong data type

Correction - float *p = new float[10];

k) int public = 1000;

Error – **Yes**, keyword can not be used as a variable name.

Correction - int public1 = 1000;

l) char name[33] = "USA";

Error – **Yes,** array size of char must be larger than the number of characters in the string.

Correction - char name[4] = "USA";

3.) Assume that a bank maintains two kinds of accounts for customers, one called a **savings account** and the other as a **current account**.

- ✓ The **savings account** provides simple interest and withdrawal facilities but no cheque book facility.
- ✓ The **current account** provides a check book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class account that stores customer name, account number and type of account. From this derive the classes cur_acct and sav_acct to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:

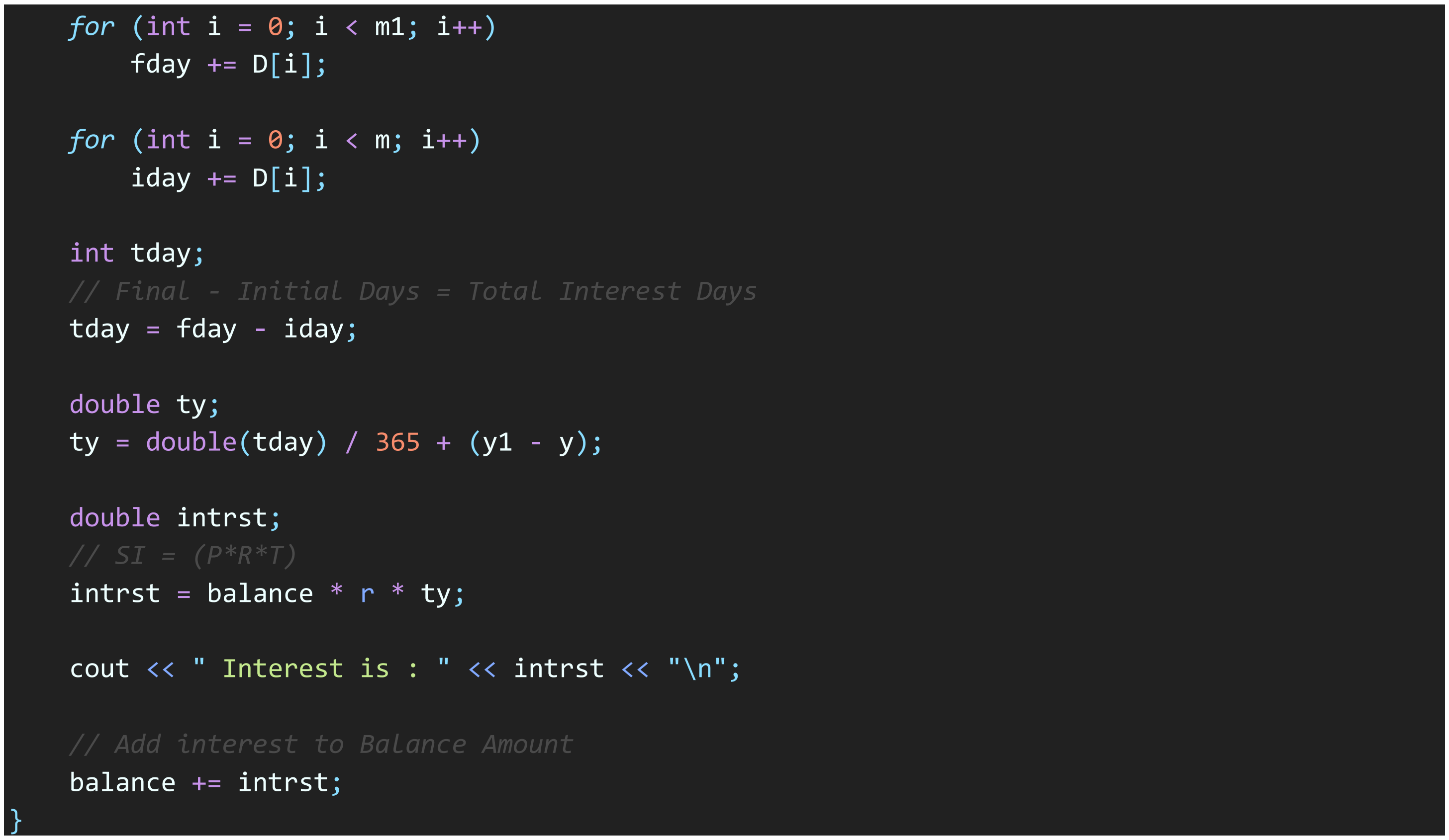
a) **Accept deposits** from a customer and update the balance.
b) **Display** the balance.
c) **Compute** and deposit interest.
d) **Permit withdrawal** and update the balance.
e) Check for the **minimum balance**, impose penalty, necessary and update the balance.
f) Do not use any constructors. Use **member functions** to initialize the class members.

Code

```
#include <iostream>
#include <string.h>
#include <string>
// [U19CS012 BHAGYA VINOD RANA]

// Miinimum Balance
#define minimum 500
// Service Charge in case if amount is less than minimum balance
#define service_charge 100
// Rate of Interest
#define r 0.10

using namespace std;

// Account Class
class account
{
protected:
```

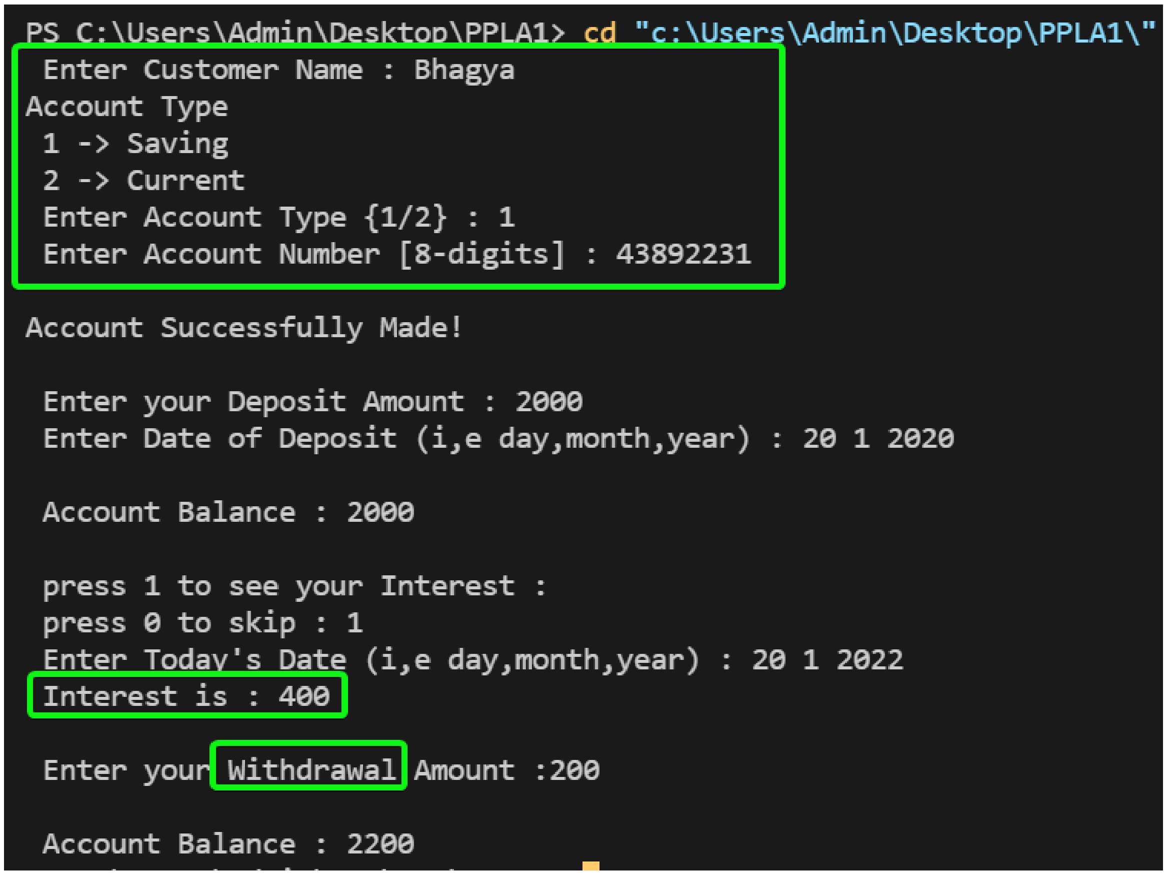
```cpp
    // Customer Name
    string name;
    // Account Number
    int ac_number;
    // Account Type
    string ac_type;

public:
    // Member Function to Create Account of type 't'
    void create_acc();
};

// Current Account Derived from Account Class
class cur_acct : public account
{
private:
    double balance;

public:
    void deposit(double d);
    void withdraw(double w);
    void display();
};

// Saving Account Derived from Account Class
class sav_acct : public account
{
    double balance;
    int d, m, y;

public:
    void deposit(double d);
    void withdraw(double w);
    void display();
    void set_date(int a, int b, int c)
    {
        d = a;
        m = b;
        y = c;
    }
    void interest();
};

// ---------------------------------------------------------------------------------------------
--

// Main Function
int main()
{
    sav_acct raju;
    raju.create_acc();

    // Accept Deposits
    double d;
    cout << " Enter your Deposit Amount : ";
    cin >> d;

    raju.deposit(d);

    raju.display();

    int t;
    cout << "\n press 1 to see your Interest : \n"
         << " press 0 to skip : ";

    cin >> t;

    if (t == 1)
        raju.interest();

    // Permit Withdrawal and update balance
    cout << "\n Enter your Withdrawal Amount :";

    double w;
    cin >> w;
    raju.withdraw(w);

    raju.display();

    return 0;
}

// ------------------------------MEMBER F(X) OF ACCOUNT CLASS------------------------------
----
// Member Function to Create Account of type 't'
void account::create_acc()
{
    cout << " Enter Customer Name : ";
    cin >> name;

    cout << "Account Type" << endl;
    cout << " 1 -> Saving\n 2 -> Current\n ";
    cout << "Enter Account Type {1/2} : ";
    int ch;
    cin >> ch;

    if (ch == 1)
        ac_type = "savings";
    else
        ac_type = "current";
    string s;
    do
    {
        cout << " Enter Account Number [8-digits] : ";
        cin >> ac_number;
        s = to_string(ac_number);
        if (s.length() != 8)
            cout << "Please Enter Valid Account Number!\n";
    } while (s.length() != 8);

    cout << "\nAccount Successfully Made!\n\n";
}

// -----------------------------MEMBER F(X) OF CURR ACCOUNT CLASS----------------------------
----
void cur_acct::deposit(double d)
{
    balance += d;
}

void cur_acct::withdraw(double w)
{
    if (balance < w)
        cout << " Sorry! Insufficient Balance!\n";
    else
    {
        balance -= w;
        if (balance < minimum)
        {
            cout << "\n Your current balance is :" << balance << " which is less than" <<
minimum << "\n your account is discharged by " << service_charge << "Rs \n"
                 << " You must store " << minimum << "Rs to avoid discharge\n "
                 << " Do you want to Withdraw ? Press 1 -> YES OR Press 0 -> NO \n"
                 << " What is your Choice ?";

            int opt;
            cin >> opt;
            if (opt == 0)
                balance += w;
        }
    }
}

void cur_acct::display()
{
    cout << "\n Account Balance = " << balance << "\n";
}
// -----------------------------MEMBER F(X) OF SAVING ACCOUNT CLASS--------------------------
----
```

```cpp
void sav_acct::deposit(double d)
{
    int x, y, z;
    cout << " Enter Date of Deposit (i,e day,month,year) : ";
    cin >> x >> y >> z;
    set_date(x, y, z);
    balance = d;
}

void sav_acct::withdraw(double w)
{
    if (balance < w)
        cout << " Sorry! Insufficient Balance!\n";
    else
    {
        balance -= w;
        if (balance < minimum)
        {
            cout << "\n Your current balance is :" << balance << " which is less than" <<
minimum << "\n your account is discharged by " << service_charge << "Rs \n"
                 << " You must store " << minimum << "Rs to avoid discharge\n "
                 << " Do you want to Withdraw ? Press 1 -> YES OR Press 0 -> NO \n"
                 << " What is your Choice ?";

            int opt;
            cin >> opt;
            if (opt == 0)
                balance += w;
        }
    }
}

void sav_acct::display()
{
    cout << "\n Account Balance : " << balance << endl;
}

void sav_acct::interest()
{
    // No of Days in Different Month of Years
    int D[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    int d1, y1, m1;
    cout << " Enter Today's Date (i,e day,month,year) : ";
    cin >> d1 >> m1 >> y1;

    int iday, fday;
    iday = d;
    fday = d1;
```



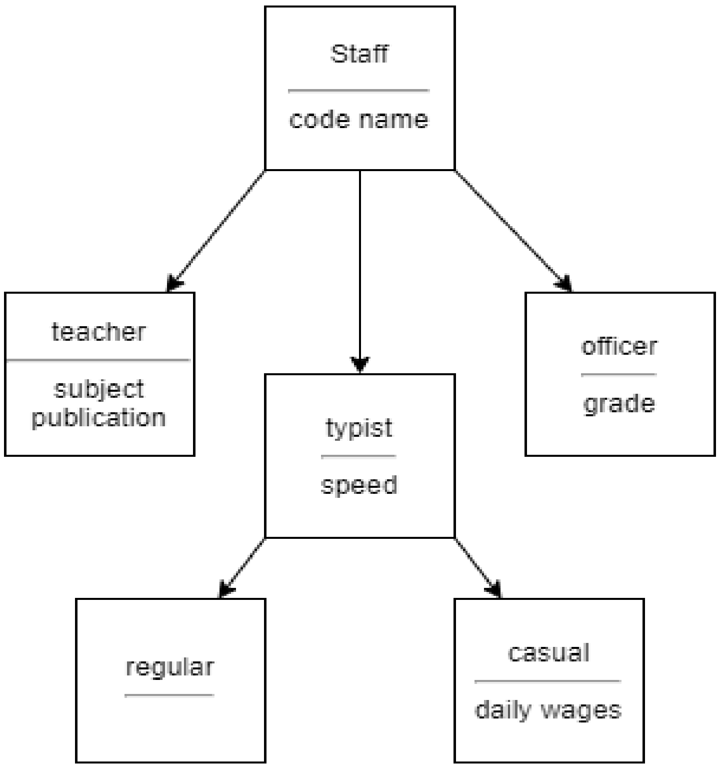
```
    for (int i = 0; i < m1; i++)
        fday += D[i];

    for (int i = 0; i < m; i++)
        iday += D[i];

    int tday;
    // Final - Initial Days = Total Interest Days
    tday = fday - iday;

    double ty;
    ty = double(tday) / 365 + (y1 - y);

    double intrst;
    // SI = (P*R*T)
    intrst = balance * r * ty;

    cout << " Interest is : " << intrst << "\n";

    // Add interest to Balance Amount
    balance += intrst;
}
```

Output



```
PS C:\Users\Admin\Desktop\PPLA1> cd "c:\Users\Admin\Desktop\PPLA1\"
 Enter Customer Name : Bhagya
Account Type
 1 -> Saving
 2 -> Current
 Enter Account Type {1/2} : 1
 Enter Account Number [8-digits] : 43892231

Account Successfully Made!

 Enter your Deposit Amount : 2000
 Enter Date of Deposit (i,e day,month,year) : 20 1 2020

 Account Balance : 2000

 press 1 to see your Interest :
 press 0 to skip : 1
 Enter Today's Date (i,e day,month,year) : 20 1 2022
 Interest is : 400

 Enter your Withdrawal Amount :200

 Account Balance : 2200
```

4.) An educational institution wishes to maintain a database of its Employees. The database is divided into a number of classes whose **hierarchical relationships** are shown in the following figure. The figure also shows the minimum information required for each class. Specify **all classes and define functions** to create the database and retrieve individual information as and when required.



The database created does not include educational information of the staff. It has been decided to add this information to **teachers and officers** (and not for typists) which will help management in decision making with regard to training, promotions etc.

Add another data class called **education** that holds two pieces of educational information namely **highest qualification** in general education and highest professional qualification. This class should be inherited by the class's teacher and officer.

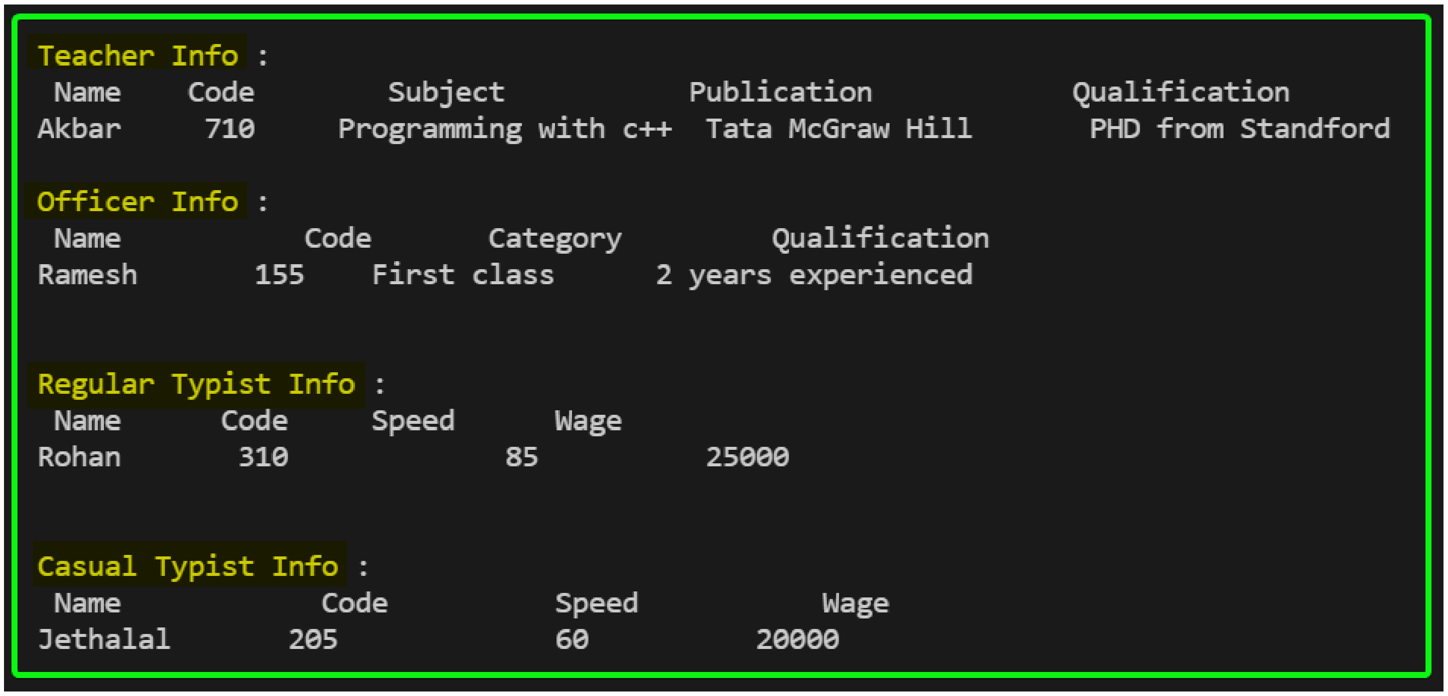
Code

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

using namespace std;
```

```cpp
// Staff Class
class staff
{
protected:
    // Staff Code & Name
    int code;
    string name;

public:
    void set_info(string n, int c)
    {
        name = n;
        code = c;
    }
};

// Education added for Staff
class education : public staff
{
protected:
    string quali;

public:
    void set_qualification(string q) { quali = q; }
};

// Teacher Class
class teacher : public education
{
protected:
    // Subject and Publication
    string sub, publication;

public:
    // To Intialize the Teacher's Details
    void set_details(string s, string p)
    {
        sub = s;
        publication = p;
    }

    // To Display the Teachers Information
    void show()
    {
        cout << " Name " << setw(8) << " Code " << setw(15)
             << " Subject " << setw(22) << " Publication "
             << setw(25) << " Qualification " << endl
             << name << setw(8) << code << setw(25)
             << sub << setw(18) << publication << setw(25) << quali << endl;
    }
};

// Officer's Class
class officer : public education
{
    // Officer Grade
    string grade;

public:
    void set_details(string g)
    {
        grade = g;
    }

    // To Display the Officers Information
    void show()
    {
        cout << " Name " << setw(15) << " Code " << setw(15) << " Category "
             << setw(22) << " Qualification " << endl
             << name << setw(10)
             << code << setw(15) << grade << setw(25) << quali << endl
             << endl;
    }
};

// Typist Class
class typist : public staff
{
protected:
    float speed;

public:
    void set_speed(float s)
    {
        speed = s;
    }
};

// Regular Typist which inherits Publicly from Typist Class
class regular : public typist
{
protected:
    float wage;

public:
    void set_wage(float w) { wage = w; }
    void show()
    {
        cout << " Name " << setw(10) << " Code " << setw(10) << " Speed "
```

```
             << setw(10) << " Wage " << endl
             << name << setw(10) << code
             << setw(15) << speed << setw(15) << wage << endl
             << endl;
    }
};

// Casual Typist which inherits Publicly from Typist Class
class causal : public typist
{
    float wage;

public:
    void set_wage(float w) { wage = w; }
    void show()
    {
        cout << " Name " << setw(16) << " Code " << setw(15) << " Speed "
             << setw(15) << " Wage " << endl
             << name << setw(10) << code
             << setw(15) << speed << setw(15) << wage << endl
             << endl;
    }
};

int main()
{
    // Teacher
    teacher t;
    t.set_info("Akbar", 710);
    t.set_details("Programming with c++", "Tata McGraw Hill");
    t.set_qualification("PHD from Standford");

    // Officer
    officer o;
    o.set_info("Ramesh", 155);
    o.set_details("First class");
    o.set_qualification("2 years experienced");

    // Regular Typist
    regular rt;
    rt.set_info("Rohan", 310);
    rt.set_speed(85);
    rt.set_wage(25000);

    // Casual Typist
    causal ct;
    ct.set_info("Jethalal", 205);
    ct.set_speed(60);
    ct.set_wage(20000);
    cout << "\nTeacher Info : " << endl;
    t.show();

    cout << "\nOfficer Info : " << endl;
    o.show();

    cout << "\nRegular Typist Info : " << endl;
    rt.show();

    cout << "\nCasual Typist Info : " << endl;
    ct.show();

    return 0;
}
```

<u>**Output**</u>



```
Teacher Info :
 Name     Code        Subject              Publication            Qualification
Akbar      710     Programming with c++  Tata McGraw Hill          PHD from Standford

Officer Info :
 Name           Code        Category           Qualification
Ramesh       155     First class       2 years experienced


Regular Typist Info :
 Name       Code      Speed       Wage
Rohan        310              85           25000


Casual Typist Info :
 Name             Code            Speed            Wage
Jethalal        205              60          20000
```

SUBMITTED BY: U19CS012

BHAGYA VINOD RANA