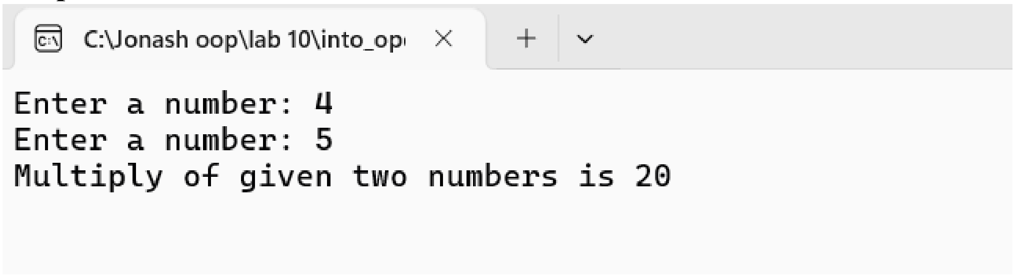
LAB 10

1. WAP to overload '*' operator

```
#include <iostream>
using namespace std;
class multiply
{
   int a;

public:
   void getdata()
   {
      cout << "Enter a number: ";
      cin >> a;
   }
   void putdata()
   {
      cout << "Multiply of given two numbers is " << a;
   }
   multiply operator*(multiply b)
   {
      multiply c;
      c.a = a * b.a;
      return c;
   }
};
int main()
{
   multiply aa, bb, cc;
   aa.getdata();
   bb.getdata();
   cc = aa * bb;
   cc.putdata();
   return 0;
}
```

Output:

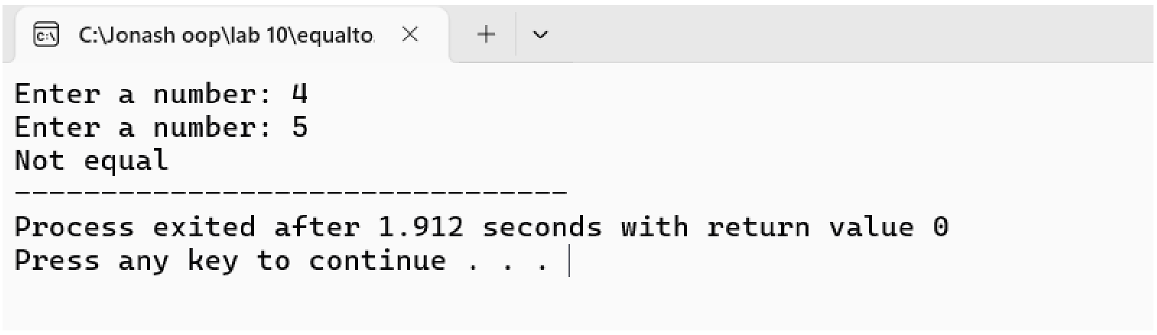


```
Enter a number: 4
Enter a number: 5
Multiply of given two numbers is 20
```

2. WAP to overload '==' operator

```
#include<iostream>
using namespace std;
class equalto
{
int x;
public:
void getdata()
{
   cout<<"Enter a number: ";
   cin>>x;
}
bool operator==(equalto b)
{
   if (x==b.x)
   return true;
   else
   return false;
}
};
int main()
{
   equalto aa,bb;
   aa.getdata();
   bb.getdata();
   if (aa==bb)
   cout<<"Both are equal";
   else
   cout<<"Not equal";
}
```

Output:



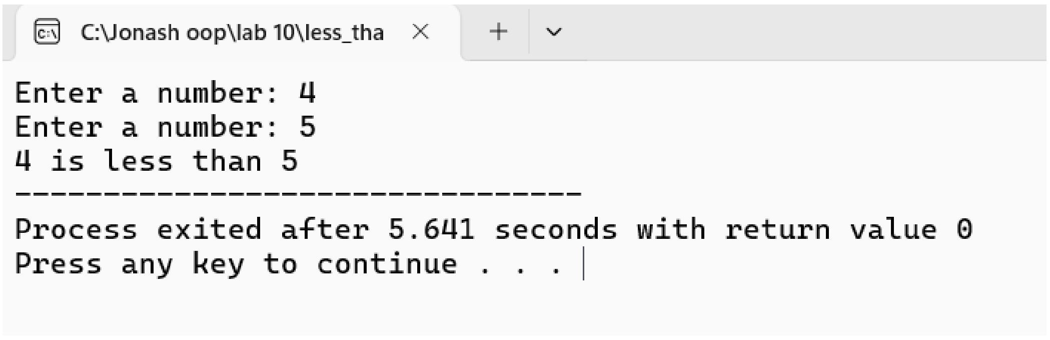
```
C:\Jonash oop\lab 10\equalto.
Enter a number: 4
Enter a number: 5
Not equal
--------------------------------
Process exited after 1.912 seconds with return value 0
Press any key to continue . . .
```

3. WAP to overload '>=' operator

```
// WAP to overload '>=' operator
#include <iostream>
using namespace std;
class demo
{
   int x;

public:
   void getdata()
   {
      cout << "Enter a number: ";
      cin >> x;
```

```
    }
    void showdata()
    {
        cout << x;
    }
    bool operator>=(demo b)
    {
        if (x >= b.x)
            return true;
        else
            return false;
    }
};
int main()
{
    demo aa, bb;
    aa.getdata();
    bb.getdata();
    aa.showdata();
    if (aa >= bb)
        cout << " is greater than or equal to ";
    else
        cout << " is less than ";
    bb.showdata();
}
```

Output:

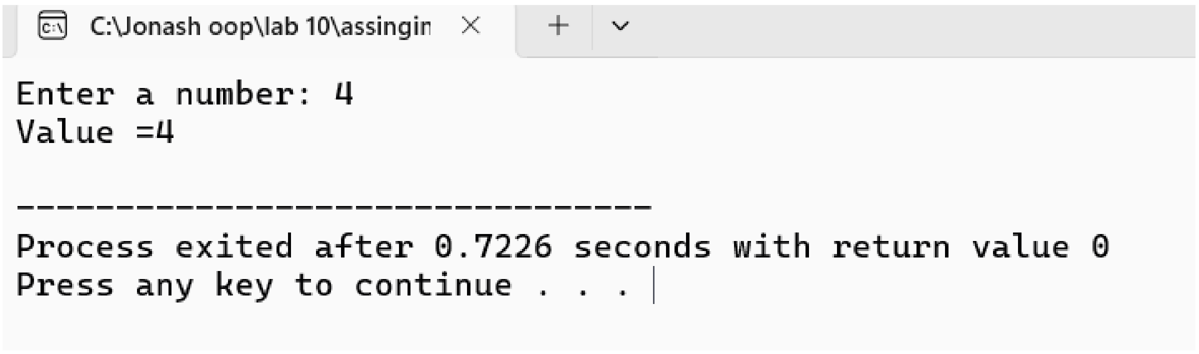


```
Enter a number: 4
Enter a number: 5
4 is less than 5
--------------------------------
Process exited after 5.641 seconds with return value 0
Press any key to continue . . .
```

4. WAP to overload '=' operator

```
#include <iostream>
using namespace std;
class demo
{
    int a;

public:
    void getdata()
    {
        cout << "Enter a number: ";
        cin >> a;
    }
    void putdata()
    {
        cout << "Value =" << a << endl;
    }
    void operator=(demo bb)
    {
        a = bb.a;
    }
};
int main()
{
    demo aa, cc;
    aa.getdata();
    cc = aa;
    cc.putdata();
    return 0;
}
```

Output:



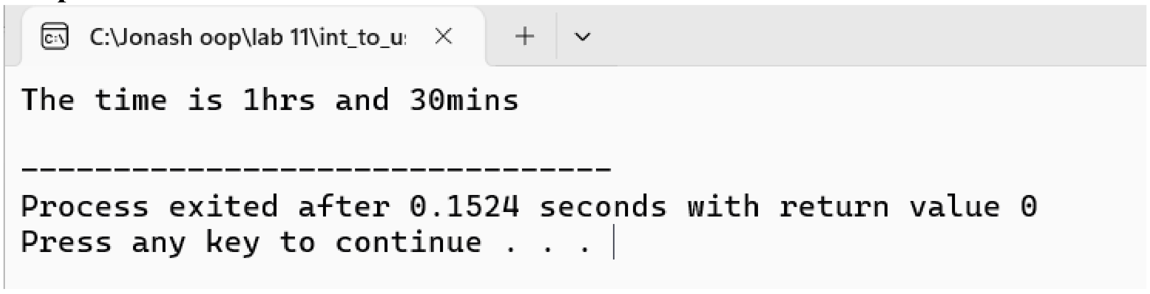
```
Enter a number: 4
Value =4

--------------------------------
Process exited after 0.7226 seconds with return value 0
Press any key to continue . . .
```

LAB 11

1. WAP to convert int to user defined type [Convert 95 minutes into hours and minutes]

```
#include <iostream>
using namespace std;
class time
{
   int hour;
   int minute;

public:
   time(int t)
   {
      hour = t / 60;
      minute = t % 60;
   }
   void display()
   {
      cout << "The time is " << hour << "hrs and " << minute << "mins" << endl;
   }
};
int main()
{
   int x = 90;
   time t(x);
   t.display();
}
```

Output:



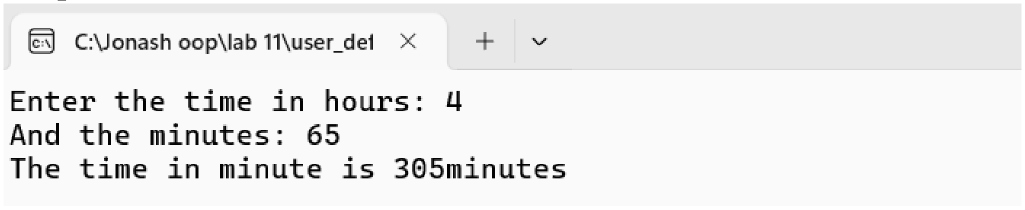
```
The time is 1hrs and 30mins

--------------------------------
Process exited after 0.1524 seconds with return value 0
Press any key to continue . . .
```

2. WAP to convert user defined to basic type. [Convert hour into minutes]

```
#include <iostream>
using namespace std;
class time
{
   int hour;
   int minute;

public:
   void get()
   {
      cout << "Enter the time in hours: ";
      cin >> hour;
      cout << "And the minutes: ";
      cin >> minute;
```

```
    }
    operator int()
    {
        return (hour * 60 + minute);
    }
};
int main()
{
    time t;
    t.get();
    int min = t;
    cout << "The time in minute is " << min << "minutes" << endl;
}
```

Output:



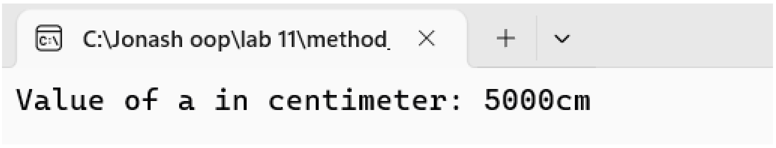
C:\Jonash oop\lab 11\user_def

```
Enter the time in hours: 4
And the minutes: 65
The time in minute is 305minutes
```

3. WAP to convert metres into cm. [Use method in destination]

```
#include <iostream>
using namespace std;
class measurement
{
private:
    int metre, centimeter;

public:
    measurement(int m)
    {
        metre = m;
        centimeter = metre * 100;
    }
    void display()
    {
        cout << "Value of a in centimeter: " << centimeter << "cm" << endl;
    }
};
int main()
{
    measurement t(50);
    t.display();
    return 0;
}
```

Output:



C:\Jonash oop\lab 11\method

```
Value of a in centimeter: 5000cm
```

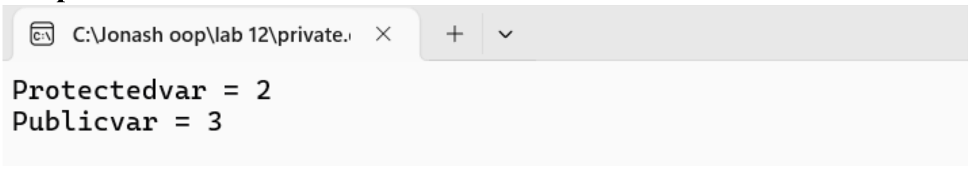
LAB 12

1. WAP to create derived class using different modes of inheritance
i) private
ii) public
iii) protected

i) Private

```
#include <iostream>
#include <conio.h>
using namespace std;
class Base
{
   int privatevar;
protected:
   int protectedvar;
public:
   int publicvar;
};

class Derived : private Base
{
public:
   void setdata()
   {
      // privatevar = 1; not accessible
      protectedvar = 2;
      publicvar = 3;
   }

   void displaydata()
   {
      cout << "Protectedvar = " << protectedvar << endl;
      cout << "Publicvar = " << publicvar << endl;
   }
};
int main()
{
   Derived obj;
   obj.setdata();
   obj.displaydata();
   // obj.publicvar=10; not accessible
   return 0;
}
```

Output:



```
Protectedvar = 2
Publicvar = 3
```

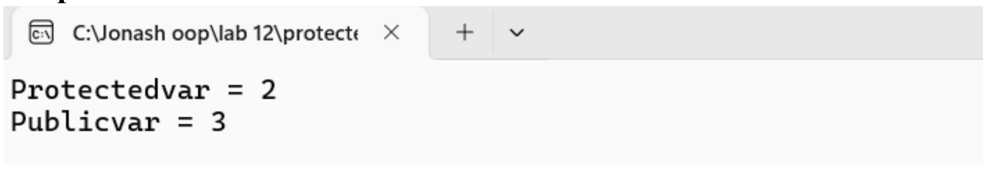
ii) Protected

```
#include <iostream>
```

LAB 12

```
#include <conio.h>
using namespace std;
class Base
{
   int privatevar;
protected:
   int protectedvar;
public:
   int publicvar;
};
class Derived : protected Base
{
public:
   void setdata()
   {
      // privatevar = 1; not accessible
      protectedvar = 2;
      publicvar = 3;
   }

   void displaydata()
   {
      cout << "Protectedvar = " << protectedvar << endl;
      cout << "Publicvar = " << publicvar << endl;
   }
};
int main()
{
   Derived obj;
   obj.setdata();
   obj.displaydata();
   // obj.publicvar=10; not accessible
   return 0;
}
```

Output:

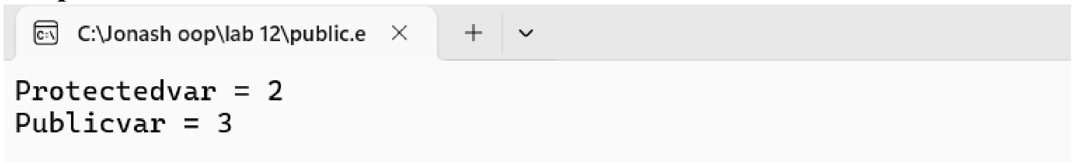

C:\Jonash oop\lab 12\protect

```
Protectedvar = 2
Publicvar = 3
```

iii) Public

```
#include <iostream>
#include <conio.h>
using namespace std;
class Base
{
   int privatevar;

protected:
    int protectedvar;

public:
    int publicvar;
};

class Derived : public Base
{
public:
    void setdata()
    {
        // privatevar = 1; not accessible
        protectedvar = 2;
        publicvar = 3;
    }

    void displaydata()
    {
        cout << "Protectedvar = " << protectedvar << endl;
        cout << "Publicvar = " << publicvar << endl;
    }
};
int main()
{
    Derived obj;
    obj.setdata();
    obj.displaydata();
    obj.publicvar = 10; // accessible
    return 0;
}
```

Output:



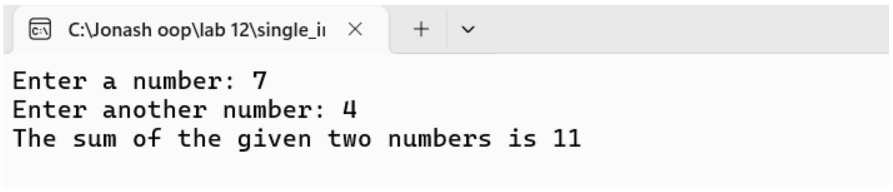
```
Protectedvar = 2
Publicvar = 3
```

2. WAP to show following
i) Single inheritance **ii) Multiple inheritance** **iii) Hierarchical inheritance**
iv) Multilevel inheritance **v) Hybrid inheritance**

i) Single inheritance

```
#include <iostream>
#include <conio.h>
using namespace std;
class A
{
protected:
    int x;
public:
    void input()
    {
        cout << "Enter a number: ";
        cin >> x;
    }
};
class B : public A
{
    int y;
public:
    void getdata()
    {
        cout << "Enter another number: ";
        cin >> y;
    }
    void sum()
    {
        cout << "The sum of the given two numbers is " << x + y << endl;
    }
};
int main()
{
    B obj;
    obj.input();
    obj.getdata();
    obj.sum();
    return 0;
}
```

Output:


C:\Jonash oop\lab 12\single_i

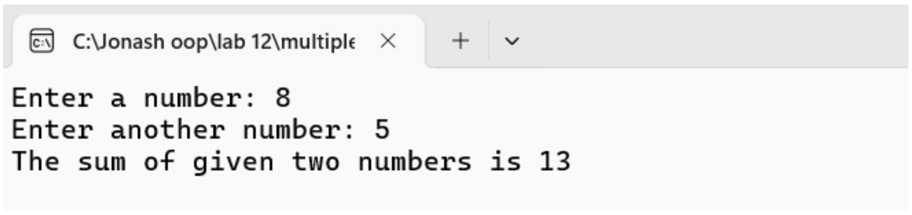
```
Enter a number: 7
Enter another number: 4
The sum of the given two numbers is 11
```

ii) Multiple inheritance

```
#include <iostream>
using namespace std;
class A
{
protected:
    int a;
public:
    void input()
    {
        cout << "Enter a number: ";
        cin >> a;
    }
};
class B
{
```

```
protected:
   int b;
   int y;
public:
   void getdata()
   {
      cout << "Enter another number: ";
      cin >> b;
   }
};
class C : public A, public B
{
public:
   void sum()
   {
      cout << "The sum of given two numbers is " << a + b << endl;
   }
};
int main()
{
   C obj;
   obj.input();
   obj.getdata();
   obj.sum();
   return 0;
}
```

Output:



C:\Jonash oop\lab 12\multiple

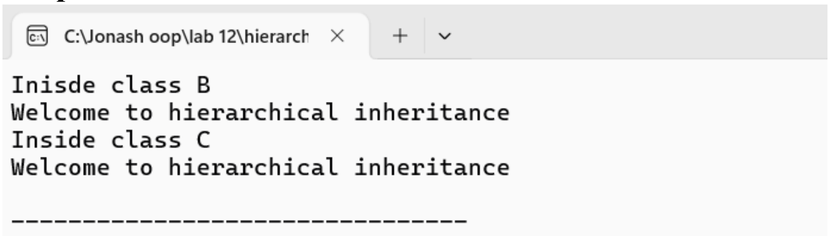
```
Enter a number: 8
Enter another number: 5
The sum of given two numbers is 13
```

iii) Hierarchical inheritance

```
#include <iostream>
using namespace std;
class A
{
public:
   void message()
   {
      cout << "Welcome to hierarchical inheritance" << endl;
   }
};
class B : public A
{
public:
   void display()
```

```
    {
        cout << "Inisde class B" << endl;
    }
};
class C : public A
{
public:
    void output()
    {
        cout << "Inside class C" << endl;
    }
};
int main()
{
    B objb;
    C objc;
    objb.display();
    objb.message();
    objc.output();
    objc.message();
    return 0;
}
```

Output:



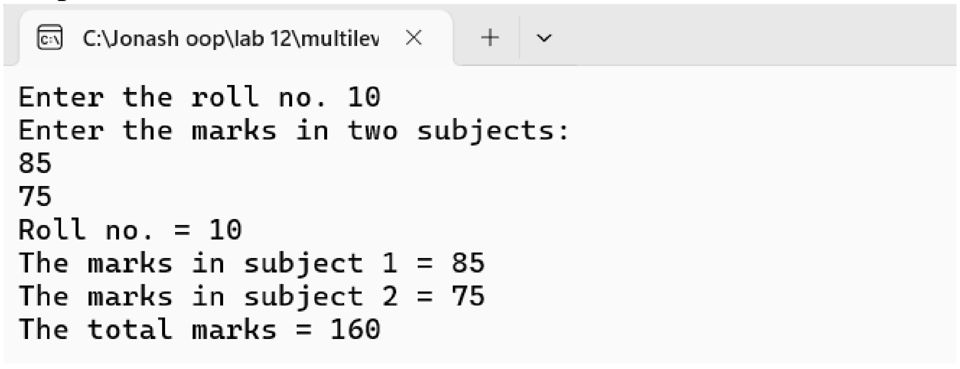
```
Inisde class B
Welcome to hierarchical inheritance
Inside class C
Welcome to hierarchical inheritance

--------------------------------
```

iv) Multilevel

```
#include <iostream>
using namespace std;
class A
{
protected:
    int roll;
public:
    void getroll()
    {
        cout << "Enter the roll no. ";
        cin >> roll;
    }
    void show_roll()
    {
        cout << "Roll no. = " << roll << endl;
    }
};
class B : public A
{
```

```
protected:
   int sub1, sub2;
public:
   void getmarks()
   {
      cout << "Enter the marks in two subjects: " << endl;
      cin >> sub1 >> sub2;
   }
   void show_marks()
   {
      cout << "The marks in subject 1 = " << sub1 << endl;
      cout << "The marks in subject 2 = " << sub2 << endl;
   }
};
class C : public B
{
public:
   void display()
   {
      show_roll();
      show_marks();
      cout << "The total marks = " << sub1 + sub2 << endl;
   }
};
int main()
{
   C obj;
   obj.getroll();
   obj.getmarks();
   obj.display();
   return 0;
}
```

Output:



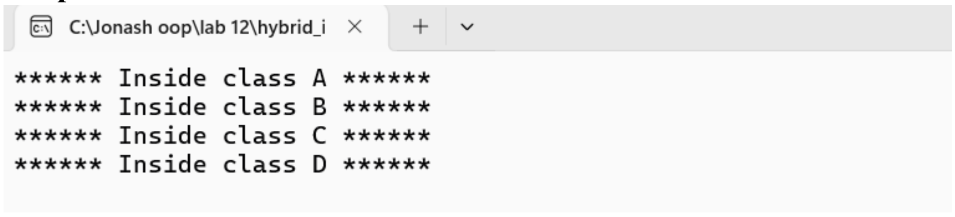
```
Enter the roll no. 10
Enter the marks in two subjects:
85
75
Roll no. = 10
The marks in subject 1 = 85
The marks in subject 2 = 75
The total marks = 160
```

v) Hybrid inheritance

```
#include <iostream>
using namespace std;
class A
{
public:
   void putdata()
```

```
    {
        cout << "****** Inside class A ******" << endl;
    }
};
class B : public A
{
public:
    void display()
    {
        cout << "****** Inside class B ******" << endl;
    }
};
class C
{
public:
    void message()
    {
        cout << "****** Inside class C ******" << endl;
    }
};
class D : public B, public C
{
public:
    void print()
    {
        cout << "****** Inside class D ******" << endl;
    }
};
int main()
{
    D aa;
    aa.putdata();
    aa.display();
    aa.message();
    aa.print();
    return 0;
}
```

Output:



```
****** Inside class A ******
****** Inside class B ******
****** Inside class C ******
****** Inside class D ******
```

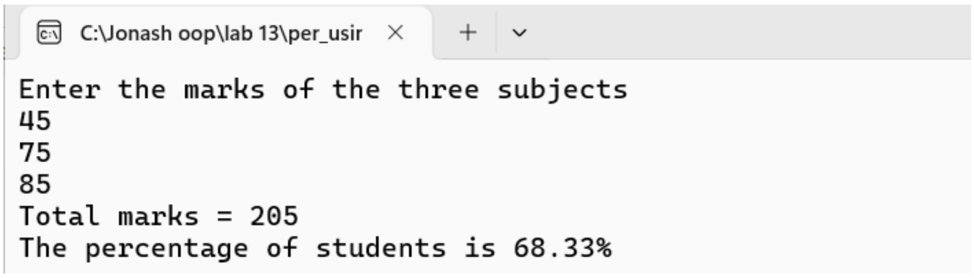
LAB 13

1. WAP to calculate percentage of student using multilevel inheritance. Accept the marks of the three students in base class. A class will be delivered from the above class which includes a function to find the total marks obtained and another class derived from this class which calculates and display the percentage of students.

```
#include <iostream>
#include <iomanip>
using namespace std;
class Marks
{
protected:
   int sub1, sub2, sub3;
public:
   void get_marks()
   {
      cout << "Enter the marks of the three subjects " << endl;
      cin >> sub1 >> sub2 >> sub3;
   }
};
class Total : public Marks
{
protected:
   int t;
public:
   void total()
   {
      t = sub1 + sub2 + sub3;
      cout << "Total marks = " << t << endl;
   }
};
class Percentage : public Total
{
   float per;
public:
   void percentage()
   {
      per = (t * 100.0) / 300;
   }
   void display()
   {
      cout << "The percentage of students is " << setprecision(2) << per << "%" << endl;
   }
};
int main(){
   Percentage aa;
   aa.get_marks();
   aa.total();
   aa.percentage();
   aa.display();
}
```

Output:

LAB 13

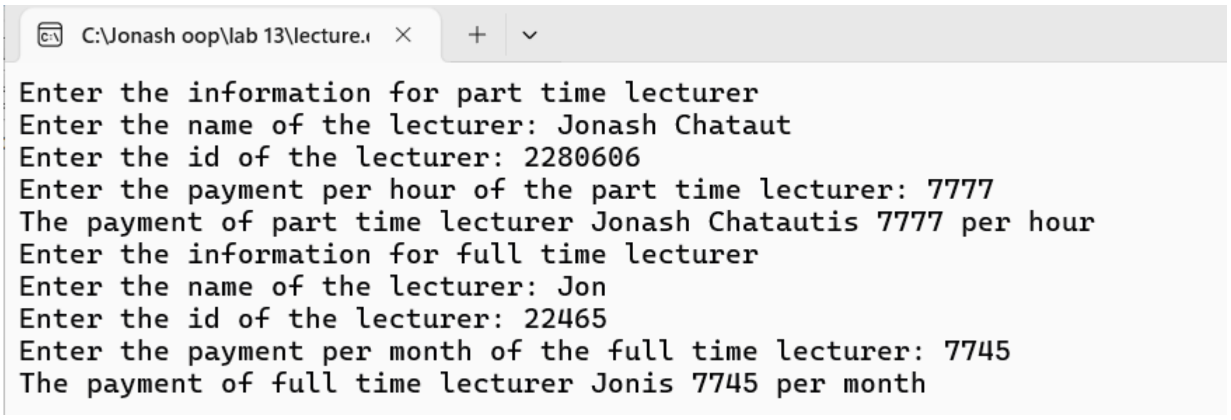

C:\Jonash oop\lab 13\per_usir

```
Enter the marks of the three subjects
45
75
85
Total marks = 205
The percentage of students is 68.33%
```

2. Create a class called lecture with id and name of lecturer. From this class derive two classes part-time which adds payperhr(float) and full-time which adds paypermonth(float). Each of the classses should have readdata() function to get data from user and printdata() function to display the data. WAP in main to test full-time and part-time by creating intance of them, asking user to fill data and display those data.

```
#include <iostream>
#include <string>
using namespace std;
class lecture
{
protected:
   int id;
   string name;

public:
   void getdata()
   {
      cout << "Enter the name of the lecturer: ";
      getline(cin, name);
      cout << "Enter the id of the lecturer: ";
      cin >> id;
   }
};
class part_time : public lecture
{
   float payperhr;

public:
   void readdata()
   {
      cout << "Enter the payment per hour of the part time lecturer: ";
      cin >> payperhr;
   }
   void printdata()
   {
      cout << "The payment of part time lecturer " << name << "is " << payperhr << " per hour" << endl;
   }
};
class full_time : public lecture
{
   float paypermonth;
public:
   void readdata()
   {
      cout << "Enter the payment per month of the full time lecturer: ";
      cin >> paypermonth;
   }
   void printdata()
   {
      cout << "The payment of full time lecturer " << name << "is " << paypermonth << " per month" <<
endl;
   }
};
int main()
{
   part_time pp;
   full_time ff;
   cout << "Enter the information for part time lecturer" << endl;
   pp.getdata();
   pp.readdata();
   pp.printdata();
   cout << "Enter the information for full time lecturer" << endl;
   cin.ignore();
   ff.getdata();
   ff.readdata();
   ff.printdata();
}
```

Output:



C:\Jonash oop\lab 13\lecture.

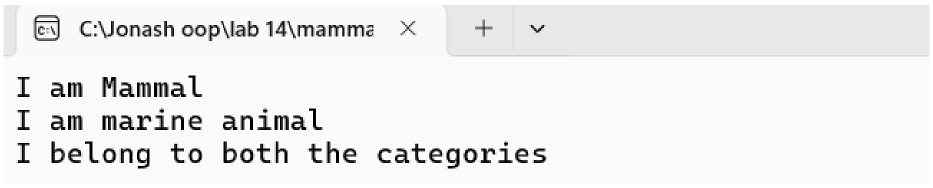
```
Enter the information for part time lecturer
Enter the name of the lecturer: Jonash Chataut
Enter the id of the lecturer: 2280606
Enter the payment per hour of the part time lecturer: 7777
The payment of part time lecturer Jonash Chatautis 7777 per hour
Enter the information for full time lecturer
Enter the name of the lecturer: Jon
Enter the id of the lecturer: 22465
Enter the payment per month of the full time lecturer: 7745
The payment of full time lecturer Jonis 7745 per month
```

LAB 14

1. Create two classes named mammals and marineanimals. Create another class named bluewhale which inherits both the above classes. Now, create function with same name on each of these classes which prints "I am Mammals", "I am marine animal" and "I belong to both the categories" respectively. Now create an object of derived class and try calling each function. What problem do you face. How would you solve this ?

```
#include <iostream>
using namespace std;
class Mammals
{
public:
   void display()
   {
      cout << "I am Mammal" << endl;
   }
};
class MarineAnimals
{
public:
   void display()
   {
      cout << "I am marine animal" << endl;
   }
};
class Bluewhale : public Mammals, public MarineAnimals
{
public:
   void display()
   {
      cout << "I belong to both the categories" << endl;
   }
};
int main()
{
   Bluewhale bb;
   bb.Mammals::display();
   bb.MarineAnimals::display();
   bb.display();
}
```

Output:



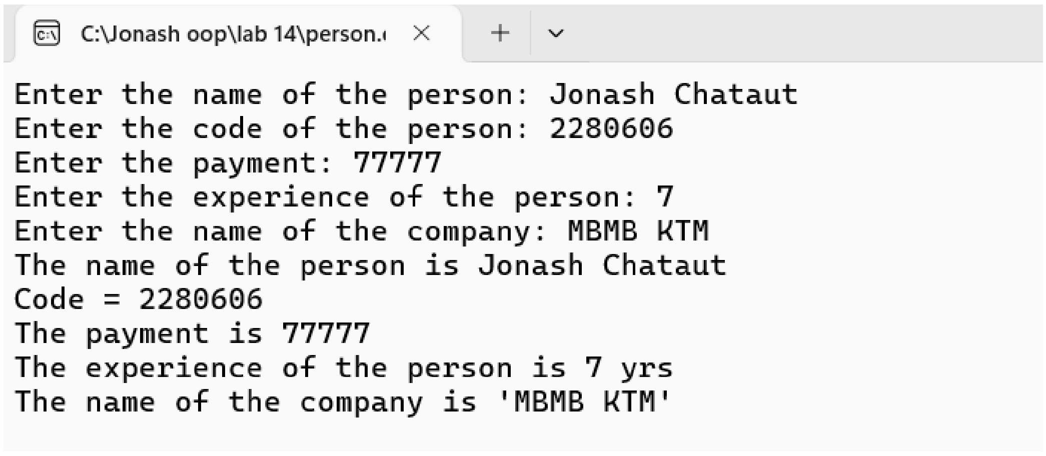
```
I am Mammal
I am marine animal
I belong to both the categories
```

2. Create a class named person with data members name and code. Derive a class named account with data member pay(float). Also derive, another class named admin with data member experience from person. Now derive class master from account and admin with data member company name. Create function to input data from user for each class. Also create function to display each data. Create object of derived master class to input and display data. What problem do you face? Solve the problem as well.

```
#include <iostream>
#include <string>
using namespace std;
class person
{
protected:
   int code;
   string name;
public:
   void input()
   {
      cout << "Enter the name of the person: ";
      getline(cin, name);
      cout << "Enter the code of the person: ";
      cin >> code;
   }
   void display()
   {
      cout << "The name of the person is " << name << endl
           << "Code = " << code << endl;
   }
};
class account : public virtual person
{
protected:
   float pay;
public:
   void get_pay()
   {
      cout << "Enter the payment: ";
      cin >> pay;
   }
   void show_pay()
   {
      cout << "The payment is " << pay << endl;
   }
};
class admin : public virtual person
{
protected:
   int exp;
public:
   void get_exp()
```

```
    {
        cout << "Enter the experience of the person: ";
        cin >> exp;
    }
    void show_exp()
    {
        cout << "The experience of the person is " << exp << " yrs" << endl;
    }
};
class master : public admin, public account
{
    string comp_name;
public:
    void get_name()
    {
        cin.ignore();
        cout << "Enter the name of the company: ";
        getline(cin, comp_name);
    }
    void show_name()
    {
        cout << "The name of the company is '" << comp_name << "'" << endl;
    }
};
int main()
{
    master aa;
    aa.input();
    aa.get_pay();
    aa.get_exp();
    aa.get_name();
    aa.display();
    aa.show_pay();
    aa.show_exp();
    aa.show_name();
}
```

Output:



```
Enter the name of the person: Jonash Chataut
Enter the code of the person: 2280606
Enter the payment: 77777
Enter the experience of the person: 7
Enter the name of the company: MBMB KTM
The name of the person is Jonash Chataut
Code = 2280606
The payment is 77777
The experience of the person is 7 yrs
The name of the company is 'MBMB KTM'
```