

NAME : Himanshu Dixit
ENROLL NO. : 21103262
BATCH : B10

Software Development Lab – II [15B17CI271]
Assignment Sheet
Week 6,7

Q1) Write a C++ program given that there are two base classes namely class A and class B from which class C is inherited. The class A contains member function `getBase()` and reads “Base” value as user input from keyboard. Class B contains member function `getHeight()` and reads “Height” value as user input from keyboard. The derived class C inherits all the public members of A and B and computes the area of the triangle.

SAMPLE OUTPUT:

enter value of base: 4.5
enter value of height: 78
area = 175.5

Solution:

```
#include <iostream>
using namespace std;
class A
{
protected:
    int base;

public:
    void getbase()
    {
        cout << "Enter value of base:";
        cin >> base;
    }
};
```

```
class B
{
protected:
    int height;

public:
    void getheight()
    {
        cout << "Enter value of height:";
        cin >> height;
    }
};
class C : public A, public B
{

public:
    void area() { cout <<"Area = "<< base * height*.5 << endl; }
};

int main()
{
    C c;
    c.getbase();
    c.getheight();
    c.area();
    return 0;
}
```

```

1  #include <iostream>
2  using namespace std;
3  class A
4  {
5  protected:
6      int base;
7
8  public:
9      void getbase()
10     {
11         cout << "Enter value of base:";
12         cin >> base;
13     }
14 };
15 class B
16 {
17 protected:
18     int height;
19
20 public:
21     void getheight()
22     {
23         cout << "Enter value of height:";
24         cin >> height;
25     }
26 };
27 class C : public A, public B
28 {
29
30 public:
31     void area() { cout << "Area = " << base * height * .5 << endl; }
32 };
33
34 int main()
35 {
36     C c;
37     c.getbase();
38     c.getheight();
39     c.area();
40     return 0;
41 }

```

```

D:\21103262_Himanshuja w5 1.exe
Enter value of base:7
Enter value of height:8
Area = 28
Process returned 0 (0x0) execution time : 13.140 s
Press any key to continue.

```

Q2) Write a C++ program, consider that there are two base classes namely class *StudentsDetails* and class *Marks* from which class *C* is inherited. The class *A* contains member function *getDetails()* that reads “students name”, “Enrollment number” value as user input from keyboard. Class *B* contains member function *getMarks()* and reads “5 subject marks” value as user input from keyboard. The derived class *C* inherits all the public members of *A* and *B* and computes the area of the triangle.

SAMPLE OUTPUT:

```

enter value of name: JOHN
enter value of eno.: JOHN123
enter value of marks [0] 89
enter value of marks [1] 78
enter value of marks [2] 67
enter value of marks [3] 86
enter value of marks [4] 57
Total = 377

```

Solution:

```

#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;
class studentdetails
{
protected:

```

```

string name;
int eno;

public:
    void getdetails()
    {
        cout << "enter name:";
        fflush(stdin);
        getline(cin, name);
        cout << "enter Eno:";
        cin>>eno;
    }
};

class marks
{
    protected:
    int marks[5];
    public:
    void getmarks(){
        cout << "enter marks of 5 subjects:";
        for(int i=0;i<5;i++){
            cin>>marks[i];
        }
    }
};

class C : public studentdetails, public marks
{
    public :
    C(){
        getdetails();
        getmarks();

        cout<<"Total="<<total();
    }
    int total(){
        return(marks[0]+marks[1]+marks[2]+marks[3]+marks[4]);
    }
}

```

```
};
```

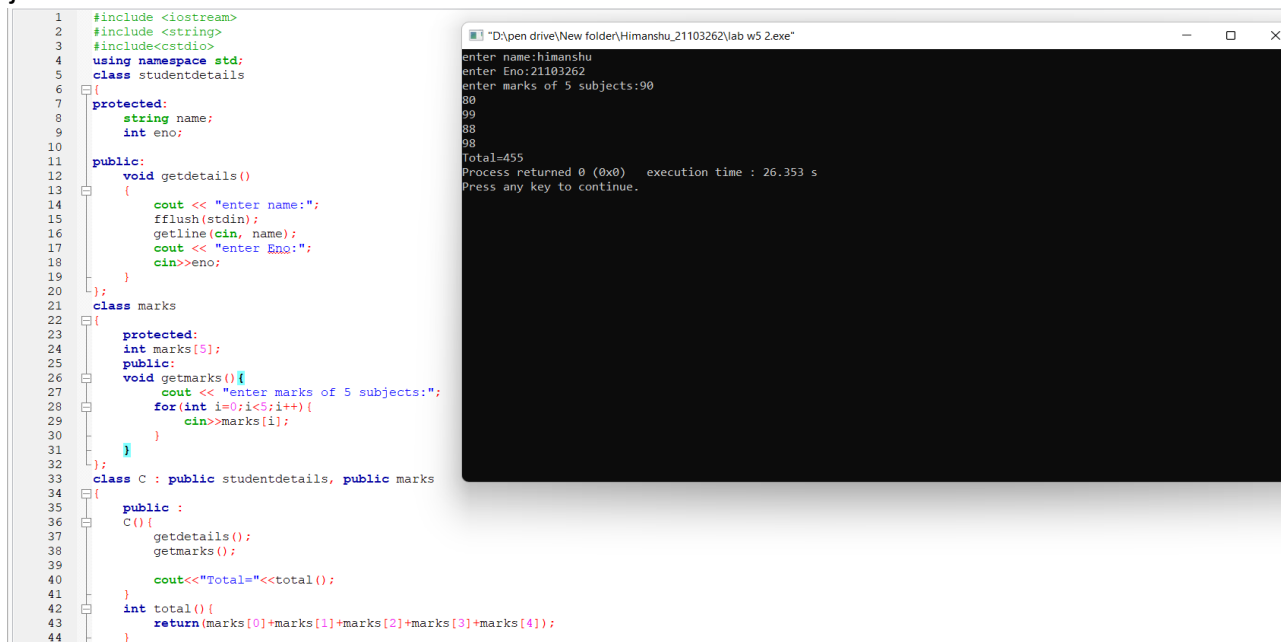
```
int main()
```

```
{
```

```
    C c;
```

```
    return 0;
```

```
}
```



The screenshot shows a C++ IDE with a code editor on the left and a console window on the right. The code defines two classes: `studentdetails` and `marks`, which are combined into class `C`. `studentdetails` has a `protected` `name` and `eno`, and a `public` `getdetails()` method. `marks` has a `protected` `marks[5]` array and a `public` `getmarks()` method. Class `C` inherits from both and has a `public` `total()` method that sums the marks. The console output shows the program running with inputs: name: himanshu, eno: 21103262, marks: 90, 80, 99, 88, 98. The total is 455, and the process returns 0 with an execution time of 26.353 s.

```
1 #include <iostream>
2 #include <string>
3 #include <cstdlib>
4 using namespace std;
5 class studentdetails
6 {
7     protected:
8         string name;
9         int eno;
10
11     public:
12         void getdetails()
13         {
14             cout << "enter name:";
15             fflush(stdin);
16             getline(cin, name);
17             cout << "enter Eno:";
18             cin >> eno;
19         }
20 };
21 class marks
22 {
23     protected:
24         int marks[5];
25     public:
26         void getmarks() {
27             cout << "enter marks of 5 subjects:";
28             for(int i=0; i<5; i++){
29                 cin >> marks[i];
30             }
31         }
32 };
33 class C : public studentdetails, public marks
34 {
35     public :
36     C() {
37         getdetails();
38         getmarks();
39         cout << "Total=" << total();
40     }
41     int total() {
42         return (marks[0]+marks[1]+marks[2]+marks[3]+marks[4]);
43     }
44 }
```

```
enter name:himanshu
enter Eno:21103262
enter marks of 5 subjects:90
80
99
88
98
Total=455
Process returned 0 (0x0)   execution time : 26.353 s
Press any key to continue.
```

Q3) Based on the virtual function concept, write the main function for the following code to display the derived class values given by user at run time.

```
#include <iostream>
```

```
using namespace std;
```

```
class base {
```

```
public:
```

```
    char fname[20];
```

```
    char surname[20];
```

```
public:
```

```
    virtual void calculate()
```

```
{
```

```
    cout << "enter fname:";
```

```
    cin >> fname;
```

```
    cout << "enter surname";
```

```

cin >> surname;
}
void display()
{
cout << "welcome" << fname << surname<<endl;
}
};
class derived : public base {
public:
void calculate()
{
cout << "enter derived_fname:";
cin>>fname;
cout << "enter derived_ surname";
cin>>surname;
}
void display()
{
cout << "welcome to derived" << fname << surname<<endl; }
};
int main()
{

```

//WRITE YOUR CODE HERE.

//Solution:

```

base *p;
p=new derived;
p->calculate();
p->display();
return 0;
}

```

Q4) Given a snippet of the program to create a base class named as base_food_Items with a virtual function named as order and total_Price . Create a derived class name Chinese. Then calculate the total_price of food items based on variables quantity and item_price.

```

#include <iostream>
using namespace std;
class base_food_items {

```

```

public:
char item_name[20];
int quantity;
int item_price;
public:
virtual void order()
{
cout << "enter item name:";
cin>> item_name;
cout << "enter quantity";
cin>> quantity;
cout << "Item price";
cin >> item_price;
}
void total_price()
{
cout<<"order is: " << item_name<<"\t"<<"quantity:"<<quantity<<endl; cout <<
"total price=" << item_price*quantity<<endl; }
};

```

Solution:

```

class chinese : public base_food_items
{
public:
    chinese()
    {
        order();
        total_price();
    }
};

```

```

int main()
{
    chinese a;
    return 0;
}

```

Q5) Write a C++ program to show the functionality of the abstract classes.

Output:

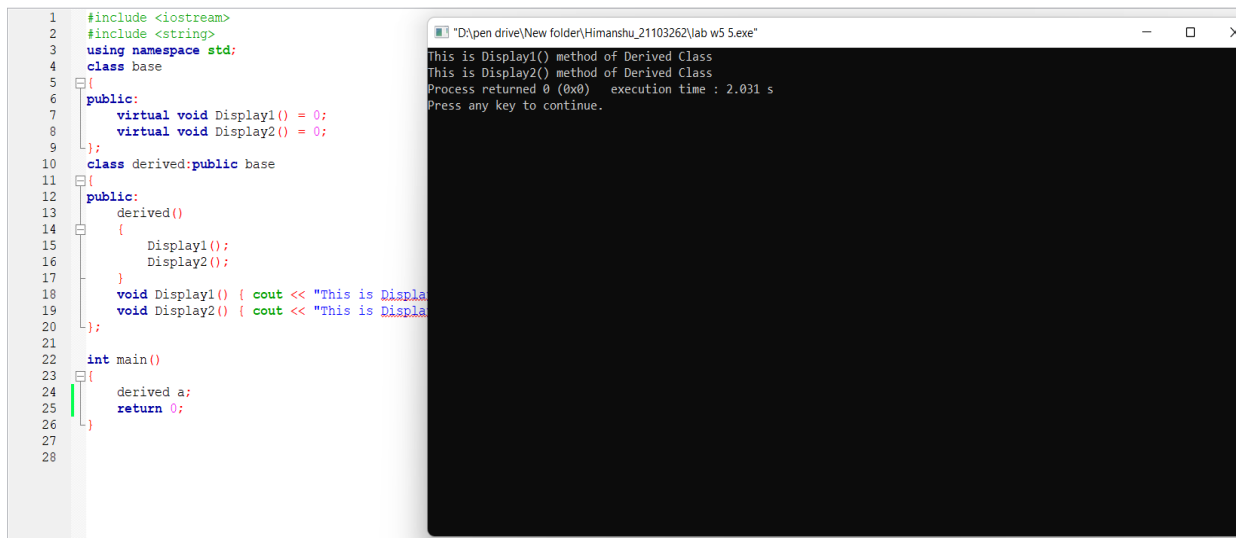
This is Display1() method of Derived Class

This is Display2() method of Derived Class

Solution:

```
#include <iostream>
#include <string>
using namespace std;
class base
{
public:
    virtual void Display1() = 0;
    virtual void Display2() = 0;
};
class derived:public base
{
public:
    derived()
    {
        Display1();
        Display2();
    }
    void Display1() { cout << "This is Display1() method of Derived Class\n"; }
    void Display2() { cout << "This is Display2() method of Derived Class"; }
};

int main()
{
    derived a;
    return 0;
}
```


The image shows a C++ program in a code editor on the left and its execution output in a terminal window on the right. The code defines a base class 'base' with two virtual functions 'Display1()' and 'Display2()' that return 0. A derived class 'derived' inherits from 'base' and overrides these functions to print "This is Display1() method of Derived Class" and "This is Display2() method of Derived Class" respectively. The 'main' function creates an object of the 'derived' class and calls both methods. The terminal window shows the output of these calls and the program's return status and execution time.

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 class base
5 {
6 public:
7     virtual void Display1() = 0;
8     virtual void Display2() = 0;
9 };
10 class derived:public base
11 {
12 public:
13     derived()
14     {
15         Display1();
16         Display2();
17     }
18     void Display1() { cout << "This is Display1() method of Derived Class"; }
19     void Display2() { cout << "This is Display2() method of Derived Class"; }
20 };
21
22 int main()
23 {
24     derived a;
25     return 0;
26 }
27
28
```

"D:\open drive\New folder\Himanshu_21103262\lab w5 5.exe"

This is Display1() method of Derived Class
This is Display2() method of Derived Class
Process returned 0 (0x0) execution time : 2.031 s
Press any key to continue.

Q6) Write a program to use constructors of the abstract class to find the sum of two numbers and display the results.

Solution:

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

```
class A
{
protected:
    int a, b;

public:
    A()
    {
        cin >> a >> b;
    }
    virtual void add() = 0;
};
```

```
class B : public A
{
public:
    void add() { cout << "Sum of a and b is:" << a + b; }
};
```

```

int main()
{
    A *b;
    b = new B;
    b->add();
    return 0;
}

```

```

1  #include <iostream>
2  using namespace std;
3
4  class A
5  {
6  protected:
7      int a, b;
8
9  public:
10     A()
11     {
12         cin >> a >> b;
13     }
14     virtual void add() = 0;
15 };
16
17 class B : public A
18 {
19 public:
20     void add() { cout << "Sum of a and b is:" << a + b; }
21 };
22 int main()
23 {
24     A *b;
25     b = new B;
26     b->add();
27
28     return 0;
29 }
30
31

```

```

6 7
Sum of a and b is:13
Process returned 0 (0x0) execution time : 15.356 s
Press any key to continue.

```

Q7 A) What is the output of the following code?

```

#include<iostream>
using namespace std;
class Base {};
class Derived: public Base {};
int main() {
    Base *base_ptr = new Derived;
    Derived *derived_ptr = dynamic_cast<Derived*>(base_ptr);
    if(derived_ptr != NULL)
        cout<<"It is working";
    else
        cout<<"cannot cast Base* to Derived*";
    return 0;
}

```

Solution:

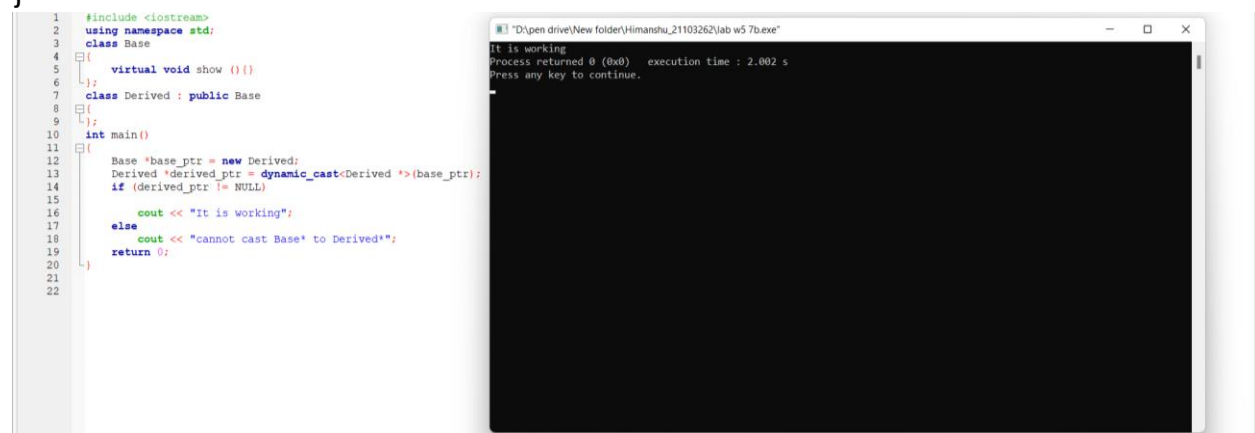
The given code will give an error due to dynamic casting of a pointer pointing to a class not containing any virtual function.

Q7 B) Rewrite the above code to rectify the error

Solution:

```
#include <iostream>
using namespace std;
class Base
{
    virtual void show (){}
};
class Derived : public Base
{
};
int main()
{
    Base *base_ptr = new Derived;
    Derived *derived_ptr = dynamic_cast<Derived *>(base_ptr);
    if (derived_ptr != NULL)

        cout << "It is working";
    else
        cout << "cannot cast Base* to Derived*";
    return 0;
}
```

The image shows a C++ code editor on the left and its output window on the right. The code editor displays the same code as the previous block, with line numbers 1 through 22. The output window, titled "D:\open drive\New folder\Himanshu_21103262\lab w5 7b.exe", shows the execution results: "It is working", "Process returned 0 (0x0) execution time : 2.002 s", and "Press any key to continue.". The code is syntactically correct because the 'Base' class now has a virtual 'show' method, which allows for dynamic casting from 'Derived' to 'Base'.