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

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

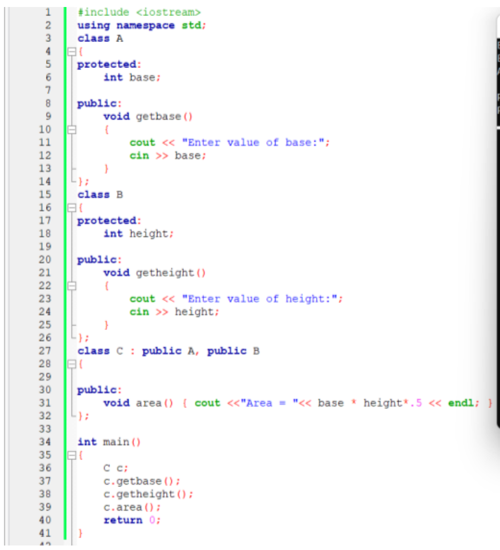
Q1) *Write a C++ program given that there are two base classes namely class A and class B from which class C is inherited. The class A contains member function getBase() and reads "Base" value as user input from keyboard. Class B contains member function getHeight() and reads "Height" value as user input 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

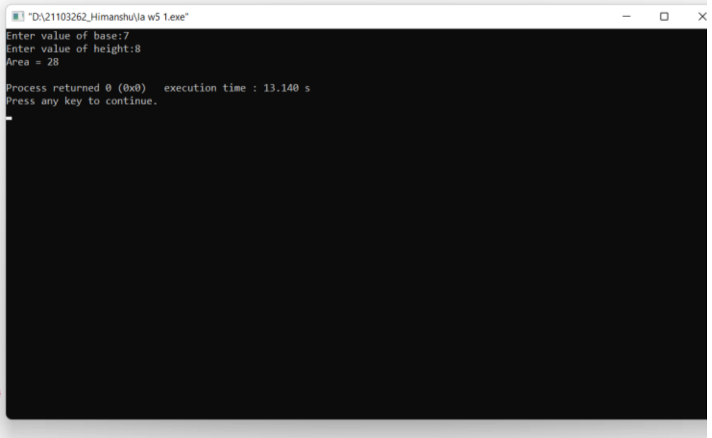
Solution:

```
#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;
}
```



```cpp
#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;
}
```



"D:\21103262_Himanshu\la 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.*

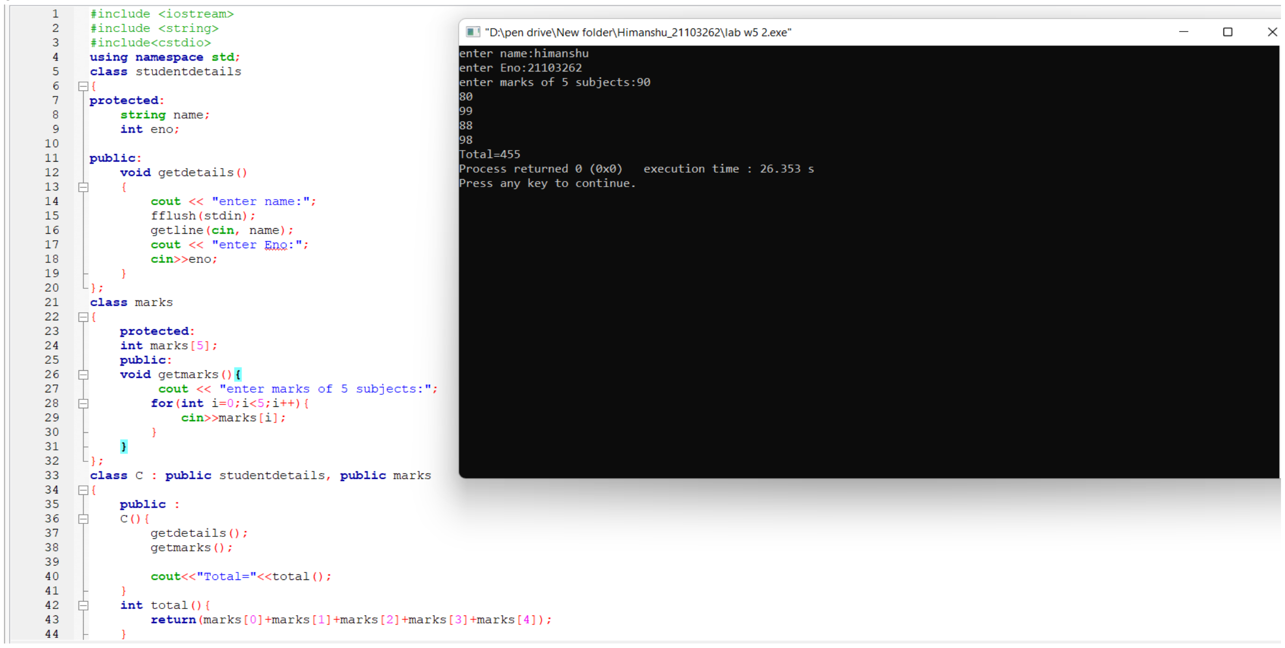
<u>SAMPLE OUTPUT:</u>

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<cstdio>
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;
}
```



```
#include <iostream>
#include <string>
#include<cstdio>
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]);
    }
```

"D:\pen drive\New folder\Himanshu_21103262\lab w5 2.exe"

```
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* *<u>base_food_</u>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.*

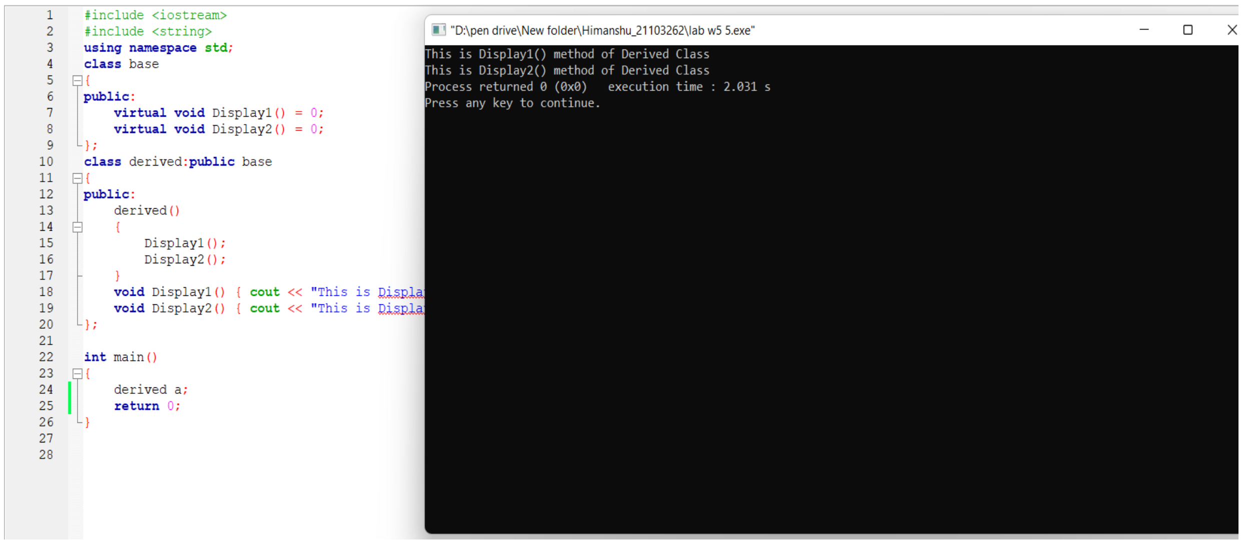
```
#include <iostream>
using namespace std;
class base_food_items {
 public:
 char item_name[20];
 int quantity;
 int item_price;
public:
 virtual void order()
 {
 cout << "enter item name:";
 cin>> item_name;
 cout << "enter quantity";
 cin>> quantity;
 cout << "Item price";
 cin >> item_price;
 }
 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;
}
```



```
#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 Displa
    void Display2() { cout << "This is Displa
};

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

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

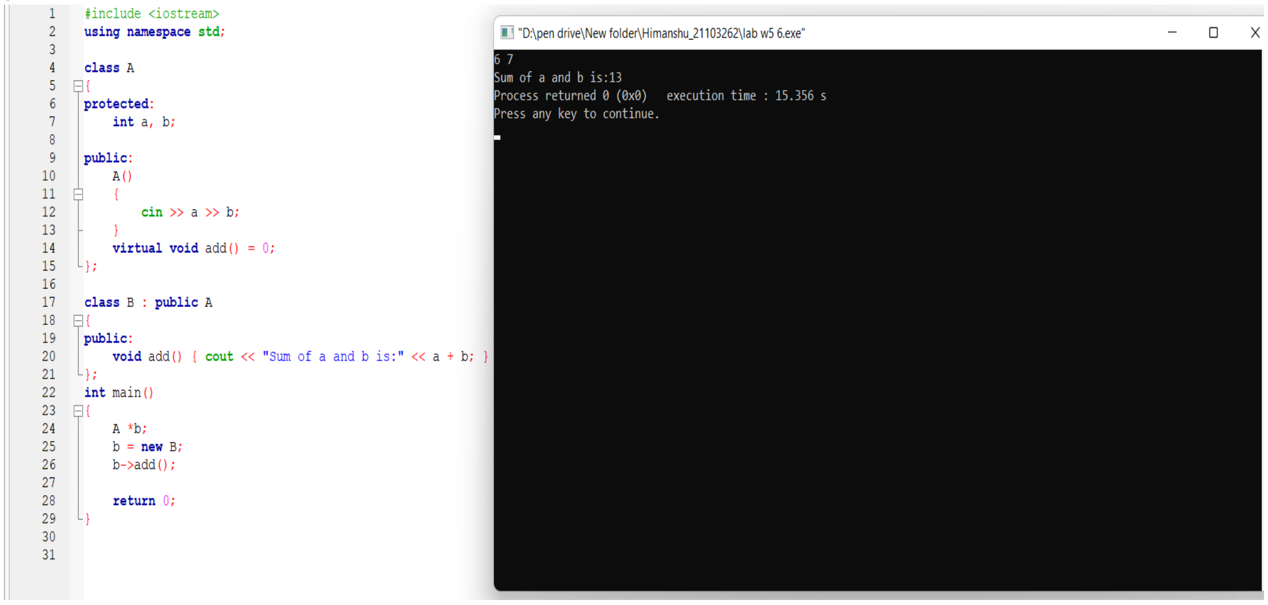
```
This is Display1() method of Derived Class
This is Display2() method of Derived Class
Process returned 0 (0x0)   execution time : 2.031 s
Press any key to continue.
```

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;
}
```



```
#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;
}
```

"D:\pen drive\New folder\Himanshu_21103262\lab w5 6.exe"

```
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?*

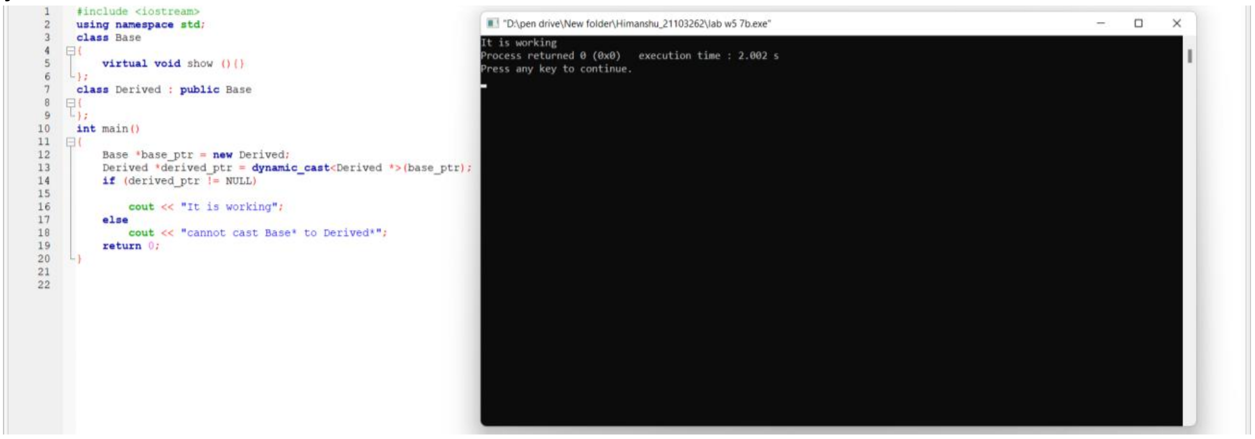
```
#include<iostream>
using namespace std;
class Base { };
class Derived: public Base {};
int main() {
 Base *base_ptr = new Derived;
 Derived *derived_ptr = dynamic_cast<Derived*>(base_ptr);
 if(derived_ptr != NULL)
 cout<<"It is working";
 else
 cout<<"cannot cast Base* to Derived*";
 return 0;
}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

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;
}
```



```
#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;
}
```

"D:\pen drive\New folder\Himanshu_21103262\lab w5 7b.exe"

```
It is working
Process returned 0 (0x0)   execution time : 2.002 s
Press any key to continue.
```