***NAME : Himanshu Dixit***

***ENROLL NO. : 21103262***

***BATCH : B10***

***Software Development fundamentals-2 [EVEN 2022]***

***Tutorial Sheet -6 (Week 6)***

***Q1. Solution:***

*D) In Derived*

*In Base*

*Because first base class pointer is pointing to derived class so derived class function is called, Next time base class pointer is pointing to base class itself so base class function is called.*

***Q2. Solution:***

*C) In Derived*

*In Derived*

*Because first base class pointer is pointing to derived class so derived class function is called. Next base class reference object is pointing to address of derived class variable, so derived class function is called.*

***Q3. Solution:***

*B) There is compile error in line “Base B”*

*Because we cannot create the object of the abstract class, i.e., the class containing the pure virtual function.*

***Q4. Solution:***

*B) Compiler error: Derived is abstract*

*As is the derived class show function is not declared. So, derived class also becomes the abstract class.*

***Q5. Solution:***

*No, the constructor cannot be a virtual.*

*No, the program will not execute due to virtual constructor is not possible.*

***Q6. Solution:***

*A) a>b*

*Using V table concept, it stores the address of Virtual function due to which the virtual class size increases. Hence, class A has more size than B.*

***Q7. Solution:***

*A)*

*Constructor: Base*

*Constructor: Derived*

*Destructor: Derived*

*Destructor: Base*

*While compiling it follows LIFO concept, first base class constructor is called than derived class, after that derived class destructor followed by base class destructor.*

***Q8. Solution:***

*C) C::fun()*

*Because base class pointer is pointing to C class, so its function is being called.*

***Q9. Solution:***

*A) In base*

*We are explicitly calling the show function of base class.*

***Q10.*** *WAP to create a abstract class shape with pure virtual functions for calculating Area () and Perimeter(). Create derive classes rectangle, square, circle to runtime override the functions Area () and Perimeter(). Use base class pointer to runtime call virtual functions.*

***Solution:***

# include <iostream>

using namespace std;

class shape

{

public:

virtual void area()=0;

virtual void peri()=0;

};

class rectangle:public shape

{

int l,b,a,p;

public:

rectangle()

{

l=5;

b=6;

}

void area()

{

a=l\*b;

cout<<"Area is :-) "<<a<<endl;

}

void peri()

{

p=2\*(l+b);

cout<<"Perimeter is :-) "<<p<<endl;

}

};

class square:public shape

{

int a, ar,p;

public:

square()

{

a=3;

}

void area()

{

ar=a\*a;

cout<<"Area is :-) "<<ar<<endl;

}

void peri()

{

p=4\*a;

cout<<"Perimeter is :-) "<<p<<endl;

}

};

class circle:public shape

{

int r;

float a,p;

public:

circle()

{

r=6;

}

void area()

{

a=3.14\*r\*r;

cout<<"Area is :-) "<<a<<endl;

}

void peri()

{

p=2\*3.14\*r;

cout<<"Perimeter is :-) "<<p<<endl;

}

};

int main()

{

shape \*ptr=new rectangle;

square s;

circle c;

ptr->area();

ptr->peri();

ptr=&s;

ptr->area();

ptr->peri();

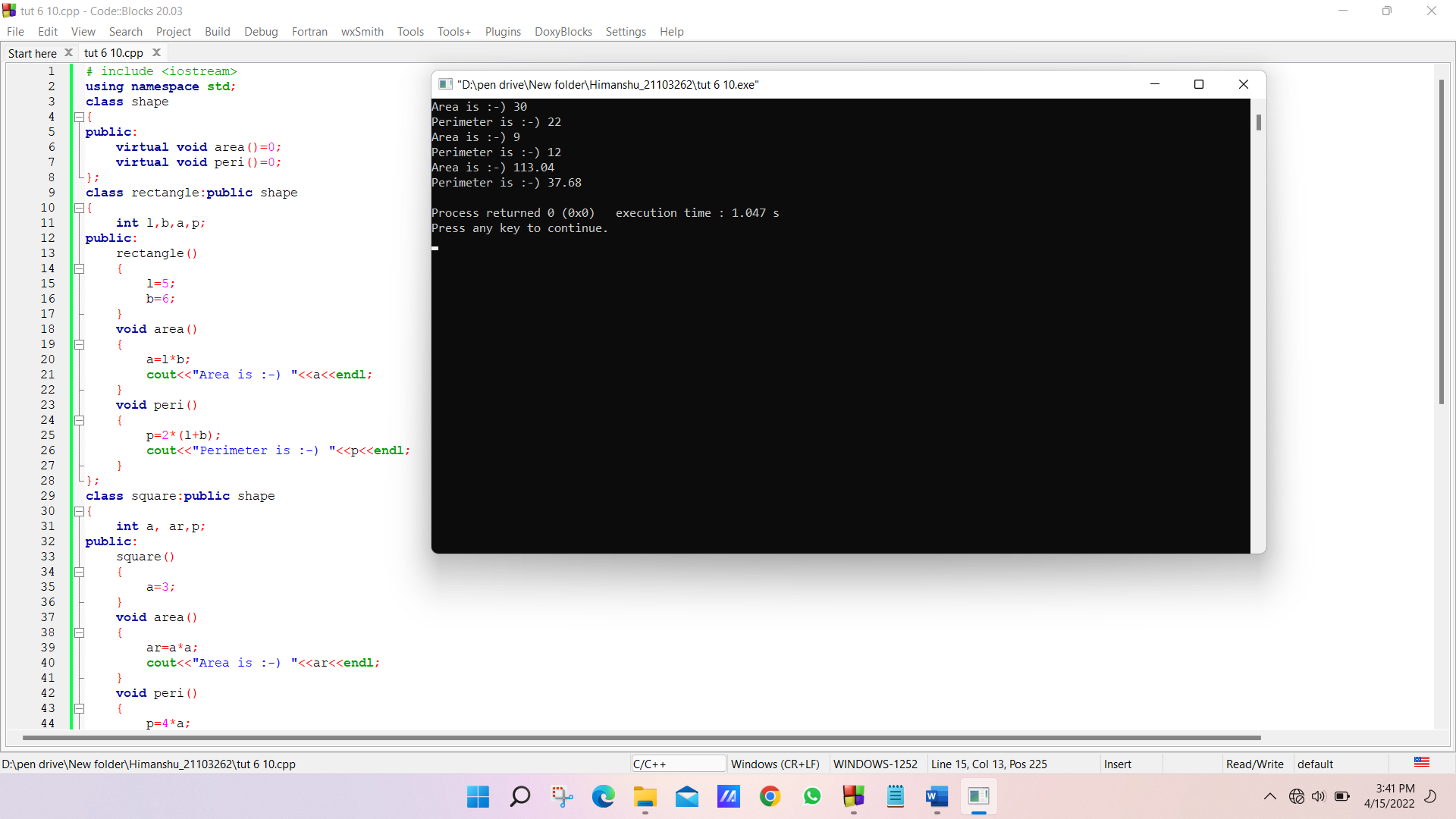
ptr=&c;

ptr->area();

ptr->peri();

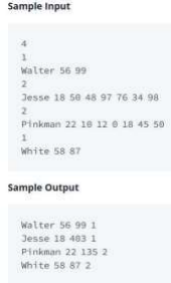
return 0;

}



***Q11.*** *Create three classes Person, Professor and Student. The class Person should have data members name and age. The classes Professor and Student should inherit from the class Person. The class Professor should have two integer members: publications and cur\_id. There will be two member functions:*

*getdata and putdata. The function getdata should get the input from the user: the name, age and publications of the professor. The function putdata should print the name, age, publications, and the cur\_id of the professor. The class Student should have two data members: marks, which is an array of size and cur\_id. It has two member functions: getdata and putdata. The function getdata should get the input from the user: the name, age, and the marks of the student in 6 subjects. The function putdata should print the name, age, sum of the marks and the cur\_id of the student. For each object being created of the Professor or the Student class, sequential id's should be assigned to them starting from 1. Solve this problem using virtual functions, constructors and static variables.*

**

***Solution:***

# include <iostream>

using namespace std;

class person

{

public:

char name[20];

int age;

person()

{

age =0;

}

virtual void getdata()=0;

virtual void putdata()=0;

};

class professor:public person

{

int publications;

static int cur\_id;

public:

professor()

{

publications=0;

cur\_id++;

}

void getdata()

{

cout<<"Enter the Name :-) ";

cin>>name;

cout<<"Enter the age :-) ";

cin>>age;

cout<<"Enter the publications :-) ";

cin>>publications;

}

void putdata()

{

cout<<"The name is :-) "<<name<<endl;

cout<<"The age is :-) "<<age<<endl;

cout<<"The publication is :-) "<<publications<<endl;

cout<<"The Current ID is :-) "<<cur\_id<<endl;

}

};

int professor::cur\_id=0;

class student:public person

{

int marks[6];

static int cur\_id;

int sum;

public:

student()

{

sum=0;

cur\_id++;

}

void getdata()

{

cout<<"Enter the Name :-) ";

cin>>name;

cout<<"Enter the age :-) ";

cin>>age;

for (int i=0; i<6; i++)

{

cout<<"Enter the marks of student in "<<i+1<<" subject :-) ";

cin>>marks[i];

sum+=marks[i];

}

}

void putdata()

{

cout<<"The name is :-) "<<name<<endl;

cout<<"The age is :-) "<<age<<endl;

cout<<"The total marks of "<<name<<" is :-) "<<sum<<endl;

cout<<"The current ID is :-) "<<cur\_id<<endl;

}

};

int student::cur\_id=0;

int main()

{

person \*ptr=new professor;

ptr->getdata();

ptr->putdata();

student s;

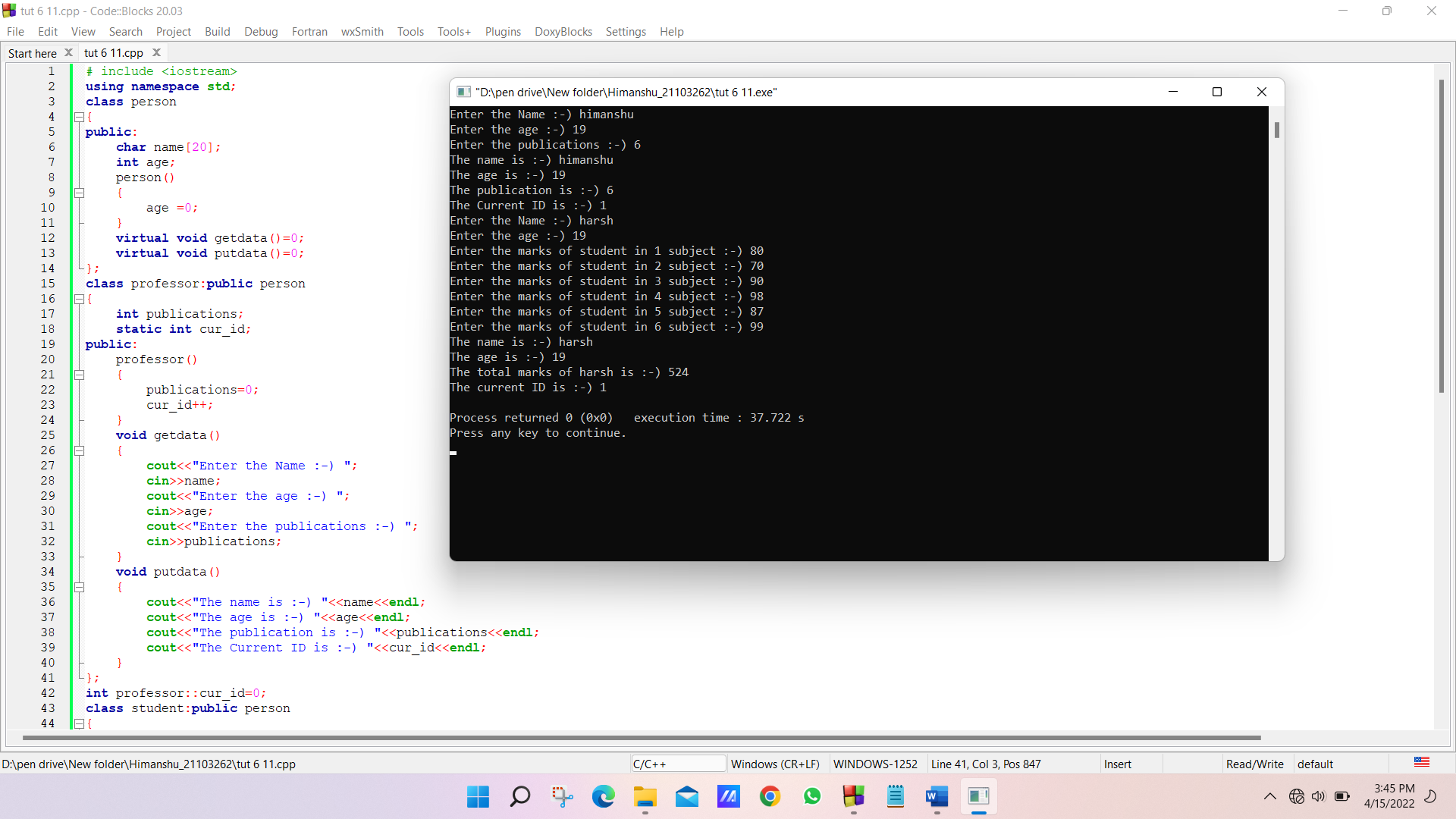
ptr= &s;

ptr->getdata();

ptr->putdata();

return 0;

}



***Q12.*** *What is Virtual destructor and explain its use?*

***Solution:*** *A virtual destructor is used to free up the memory space allocated by the derived class object or instance while deleting instances of the derived class using a base class pointer object.*

*When an object in the class goes out of scope or the execution of the main () function is about to end, a destructor is automatically called into the program to free up the space occupied by the class destructor function. When a pointer object of the base class is deleted that points to the derived class, only the parent class destructor is called due to the early bind by the compiler. In this way, it skips calling the derived class destructor, which leads to memory leaks issue in the program.*

*And when we use virtual keyword preceded by the destructor tilde (~) sign inside the base class, it guarantees that first the derived class destructor is called. Then the base class destructor is called to release the space occupied by both destructors in the inheritance class.*