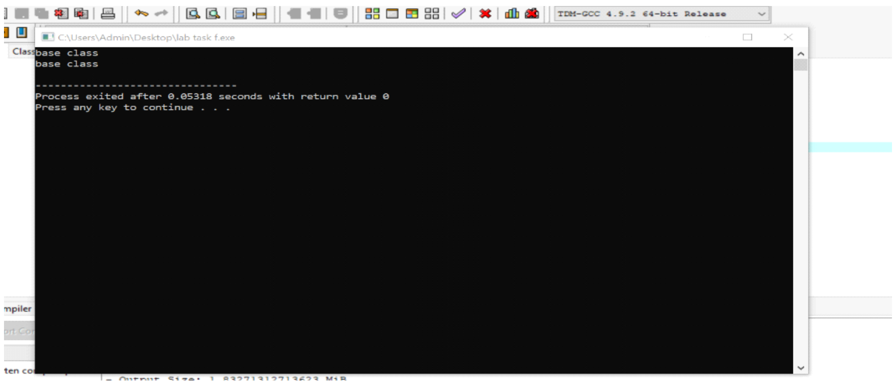
LAB REPORT

LAB TASK 1


TDM-GCC 4.9.2 64-bit Release
C:\Users\Admin\Desktop\lab task f.exe

```
base class
base class

--------------------------------
Process exited after 0.05318 seconds with return value 0
Press any key to continue . . .
```

```
#include<iostream>
            using std::cout;
            using std::endl;
             class base {
                        public:
                                    void testfunction();
             };
             class derived : public base{
                        public:
                                    void testfunction();
                                     };


                        void base::testfunction(){
                                    cout<<"base class"<<endl;
                        }
```

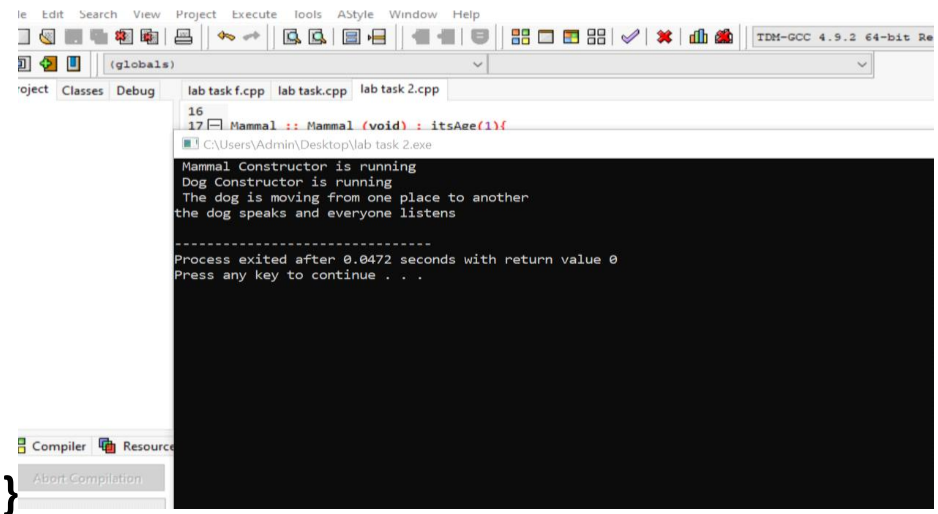
```
                    void derived::testfunction(){
                              cout<<"derived class"<<endl;
                    }

                    int main(void){
                              base* ptr = new base;
                              ptr->testfunction();
                              delete ptr;
                              ptr = new derived;
                              ptr -> testfunction();


                              delete ptr;
                              return 0;
                              }
```

LAB TASK 2

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

 class Mammal{
          public:
                    Mammal(void);
                    ~Mammal(void);
                    virtual void Move () const ;

                    virtual void Speak () const ;


                    protected :

                              int itsAge;

 };


Mammal :: Mammal (void) : itsAge(1){

          cout<<" Mammal Constructor is running "<<endl;


}


Mammal :: ~Mammal (void){

          cout<<" Mammal Destructor is running "<<endl;


}


void Mammal :: Move() const{

cout<<" Mammal moves a step! "<<endl;


}


void Mammal :: Speak() const{

cout<<"the dog speaks and everyone listens "<<endl;


}
```

```
class Dog : public Mammal{
            public:
                        Dog(void);
                        ~Dog(void);
                        virtual void Bark () const;
                         void Move () const;

                         protected:
                                    int itsAge;

};

Dog :: Dog(void) : itsAge(2){
            cout<<" Dog Constructor is running "<<endl;

}

Dog :: ~Dog (void){
            cout<<" Dog Destructor is running "<<endl;

}

void Dog :: Move() const{
   cout<<" The dog is moving from one place to another "<<endl;

}
```

```
void Dog :: Bark () const {
   cout<<" Dog is barking "<<endl;

}

int main(){
          Mammal *pDog = new Dog;

          pDog -> Move();
          pDog -> Speak ();

          return 0;
```



le Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Re
(globals)
roject Classes Debug
lab task f.cpp lab task.cpp lab task 2.cpp
Compiler Resource
Abort Compilation

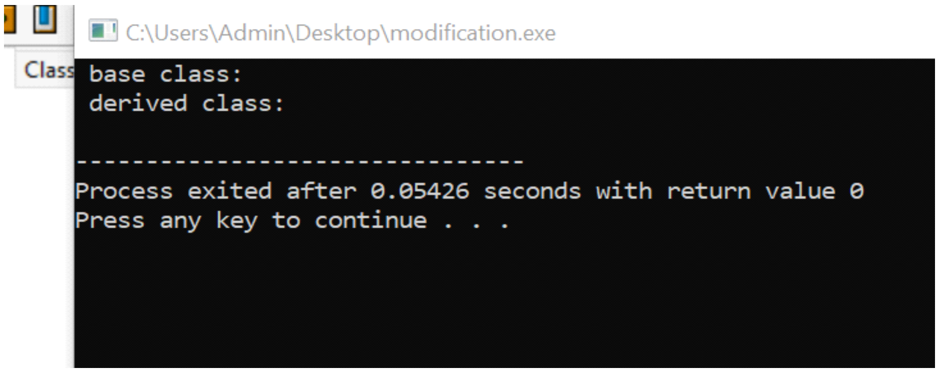
```
16
17   Mammal :: Mammal (void) : itsAge(1){
```

C:\Users\Admin\Desktop\lab task 2.exe

```
 Mammal Constructor is running
 Dog Constructor is running
 The dog is moving from one place to another
the dog speaks and everyone listens

--------------------------------
Process exited after 0.0472 seconds with return value 0
Press any key to continue . . .
```

```
}
```

TASK 1 MODIFICATION CODE :

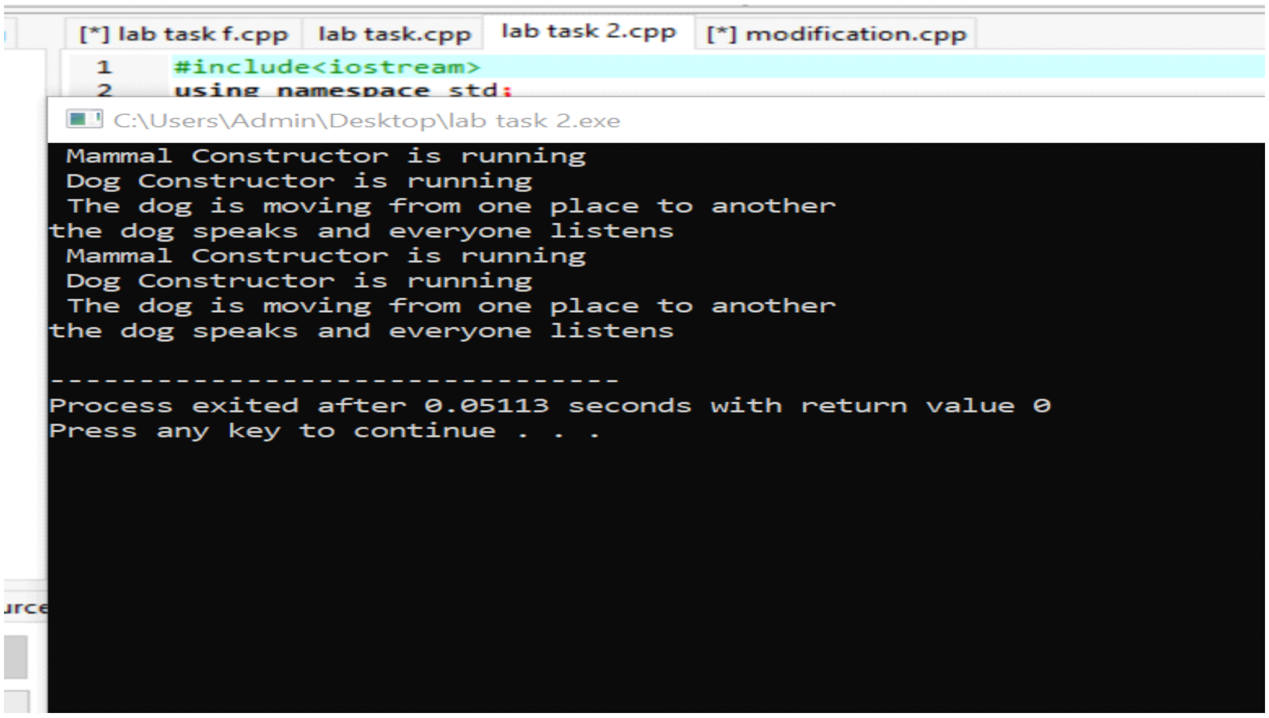


```
 base class:
 derived class:

--------------------------------
Process exited after 0.05426 seconds with return value 0
Press any key to continue . . .
```

CODE:

```
#include<iostream>
using namespace std;
        class Base {
               public:
                       virtual void testfunction();
        };
        class Derived : public Base{
               public:
                       void testfunction();
                        };

               void Base::testfunction(){
                       cout<<" base class: "<<endl;
               }
               void Derived::testfunction(){
                       cout<<" derived class: "<<endl;
               }

               int main(void){
                       Base* ptr = new Base;
                       ptr->testfunction();
                       delete ptr;
                       ptr = new Derived;
                       ptr -> testfunction();

                       delete ptr;
                       return 0;
                       }
```

TASK 2 ADD ANOTHER POINTER:



```
[*] lab task f.cpp   lab task.cpp   lab task 2.cpp   [*] modification.cpp
#include<iostream>
using namespace std;
```

```
C:\Users\Admin\Desktop\lab task 2.exe
 Mammal Constructor is running
 Dog Constructor is running
 The dog is moving from one place to another
the dog speaks and everyone listens
 Mammal Constructor is running
 Dog Constructor is running
 The dog is moving from one place to another
the dog speaks and everyone listens

--------------------------------
Process exited after 0.05113 seconds with return value 0
Press any key to continue . . .
```

CODE:

```
int main(){
        Mammal *pDog = new Dog;
        pDog -> Move();
        pDog -> Speak ();

        Mammal *pDog2 = new Dog;

        pDog -> Move();
        pDog -> Speak ();
        return 0;
}
```

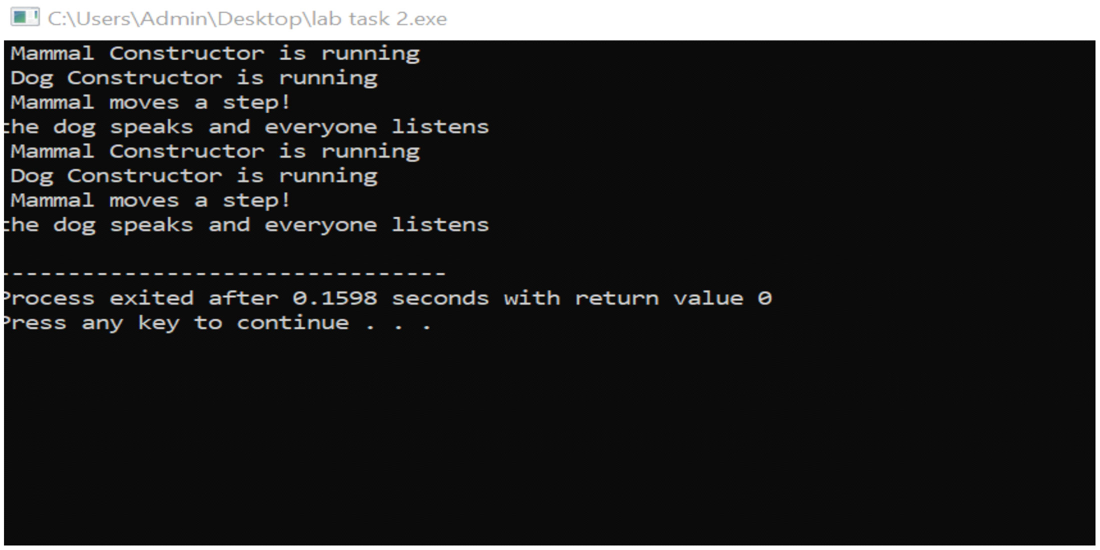
TASK 2 WITHOUT USE VIRTUAL WORD:

CODE:

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

 class Mammal{
        public:
            Mammal(void);
            ~Mammal(void);

            void Move () const ;
             void Speak () const ;

            protected :
                  int itsAge;
 };


Mammal :: Mammal (void) : itsAge(1){
              cout<<" Mammal Constructor is running "<<endl;


}


Mammal :: ~Mammal (void){
              cout<<" Mammal Destructor is running "<<endl;


}


void Mammal :: Move() const{
cout<<" Mammal moves a step! "<<endl;


}


void Mammal :: Speak() const{
cout<<"the dog speaks and everyone listens "<<endl;


}


class Dog : public Mammal{
              public:
                     Dog(void);
                     ~Dog(void);
                     virtual void Bark () const;
```

```
                    void Move () const;

                    protected:
                         int itsAge;

};

Dog :: Dog(void) : itsAge(2){
          cout<<" Dog Constructor is running "<<endl;

}

Dog :: ~Dog (void){
          cout<<" Dog Destructor is running "<<endl;

}

void Dog :: Move() const{
   cout<<" The dog is moving from one place to another "<<endl;

}

void Dog :: Bark () const {
   cout<<" Dog is barking "<<endl;

}
```

```
int main(){
        Mammal *pDog = new Dog;

        pDog -> Move();
        pDog -> Speak ();

        Mammal *pDog2 = new Dog;

        pDog -> Move();
        pDog -> Speak ();
        return 0;
}
```

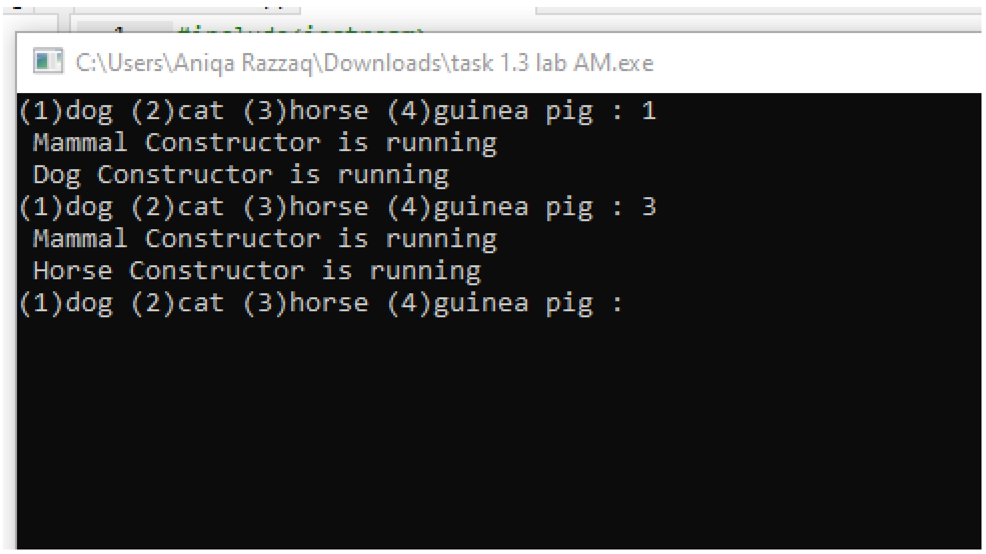


C:\Users\Admin\Desktop\lab task 2.exe

```
 Mammal Constructor is running
 Dog Constructor is running
 Mammal moves a step!
the dog speaks and everyone listens
 Mammal Constructor is running
 Dog Constructor is running
 Mammal moves a step!
the dog speaks and everyone listens

--------------------------------
Process exited after 0.1598 seconds with return value 0
Press any key to continue . . .
```

TASK 3 :

CODE:


C:\Users\Aniqa Razzaq\Downloads\task 1.3 lab AM.exe

```
(1)dog (2)cat (3)horse (4)guinea pig : 1
 Mammal Constructor is running
 Dog Constructor is running
(1)dog (2)cat (3)horse (4)guinea pig : 3
 Mammal Constructor is running
 Horse Constructor is running
(1)dog (2)cat (3)horse (4)guinea pig :
```

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

 class Mammal{
           public:
               Mammal(void);
               ~Mammal(void);

               void Move () const ;
               void Speak () const ;



               protected :
                      int itsAge;
 };

Mammal :: Mammal (void) : itsAge(1){
           cout<<" Mammal Constructor is running "<<endl;
}


Mammal :: ~Mammal (void){
                cout<<" Mammal Destructor is running "<<endl;


}


void Mammal :: Move() const{
cout<<" Mammal moves from one place to other "<<endl;


}


void Mammal :: Speak() const{
cout<<"Mammal speaks and everyone listens "<<endl;


}
class Dog : public Mammal{
                public:
                       Dog(void);
                       ~Dog(void);
                       virtual void Bark () const;
                        void Move () const;


                        protected:
                                   int itsAge;


};
```

```
Dog :: Dog(void) : itsAge(2){
                cout<<" Dog Constructor is running "<<endl;

}

Dog :: ~Dog (void){
                cout<<" Dog Destructor is running "<<endl;

}

void Dog :: Move() const{
   cout<<"Dog move from one place to other "<<endl;

}

void Dog :: Bark () const {
   cout<<" Dog is barking "<<endl;

}

class Cat : public Mammal{
public:
   Cat(void);
   ~Cat(void);
   virtual void Meow () const;
   virtual void Move () const;
```

```
protected:
    int itsAge;


};


Cat :: Cat(void) : itsAge(3){
    cout<<" Cat Constructor is running "<<endl;


}


Cat :: ~Cat (void){
    cout<<" Cat Destructor is running "<<endl;


}


void Cat :: Move() const{
    cout<<" Cat move from one place to other "<<endl;
}


void Cat :: Meow () const {
    cout<<" Cat is meowing "<<endl;
}



class Horse : public Mammal{
public:
 Horse(void);
    ~Horse(void);
    virtual void Neigh () const;
    virtual void Move () const;

protected:
    int itsAge;

};

Horse :: Horse(void) : itsAge(4){
    cout<<" Horse Constructor is running"<<endl;

}

Horse :: ~Horse (void){
    cout<<" Horse Destructor is running"<<endl;

}

void Horse :: Move() const{
    cout<<" Horse moves from one place to other "<<endl;
}

void Horse :: Neigh () const {
    cout<<" Horse is neighing "<<endl;
}


class GuineaPig : public Mammal{
```

```
public:
   GuineaPig(void);
   ~GuineaPig(void);
   virtual void Weep () const;
   virtual void Move () const;

protected:
   int itsAge;

};

GuineaPig :: GuineaPig(void) : itsAge(5){
   cout<<" GuineaPig  constructor is running "<<endl;

}

GuineaPig :: ~GuineaPig (void){
   cout<<" GuineaPig Destructor is running"<<endl;

}

void GuineaPig :: Move() const{
   cout<<" GuineaPig moves from one place to oher "<<endl;
}

void GuineaPig :: Weep () const {
   cout<<" GuineaPig is weeping "<<endl;
}
```

```
int main(){
  Mammal *theArray[5];
  Mammal *ptr;
  int choice,i;
  for(i=0; i<5 ; i++){
                cout<<"(1)dog (2)cat (3)horse (4)guinea pig : ";
                cin>> choice ;
                switch(choice){
                        case 1 : ptr = new Dog ;
                        break;
                        case 2 : ptr = new Cat ;
                        break;
                        case 3 : ptr = new Horse ;
                        break;
                        case 4 : ptr = new GuineaPig ;
                        break;
                        default : ptr = new Mammal ;
                        break ;
                  }
                  theArray[i]=ptr;


  }
  for(i=0;i<5;i++)
                theArray[i] -> Speak();


                for(i=0;i<5;i++)
     delete theArray[i];
```

```
    return 0;
}
```

CONCEPTUAL QUESTION ANSWER

Q1. If, in the example above, Mammal overrides a function in Animal, which does Dog get, the original or the overridden function?

Ans. In the above example, Mammal overrides a function in Animal so the Dog gets the overridden function.

Q2. Can a derived class make a public base function private?

Ans. Yes, and it remains private for all subsequent derivation. When a base class is privately inherited by the derived class, public members of the base class becomes the private members of the derived class and therefore, the public members of the base class can only be accessed by the member functions of the derived class. They are inaccessible to the objects of the derived class.

Q3. Why not make all class functions virtual?

Ans. Because a function only needs to be virtual iff a derived class will implement that function in a different way.

Q4. If a function (SomeFunc()) is virtual in a base class and is also overloaded, so as to take either an integer or two integers, and the derived class overrides the form taking one integer, what is called when a pointer to a derived object calls the two-integer form?

Ans. The overriding of the one-int form hides the entire base class function, and thus you will get a compile error complaining that that function requires only one int.

Here are some more questions

Q1. What is a v-table?

Ans. The v-table component is a simple wrapper component around the <table> element. Inside the component you can use all the regular table elements such as <thead>, <tbody>, <tr>, etc.

Q2. What is a virtual destructor?

Ans. Virtual destructors are useful when you might potentially delete an instance of a derived class through a pointer to base class.

Q3. How do you show the declaration of a virtual constructor?

Ans. When return a pointer, compiler needs to know the concrete type and constructor, so forward declare is not enough. What you could do is: in header file: forward declare SubVirt and CreateClass function cpp file: include MyVirt.h and define CreateClass function.

Q4. How can you create a virtual copy constructor?

Ans. A virtual copy constructor is a way to create a copy of an object using a pointer or reference to its base class, while preserving the object's dynamic type.

Q5. How do you invoke a base member function from a derived class in which you've overridden that function?

Ans. To access the overridden function of the base class, we use the scope resolution operator :: . We can also access the overridden function by using a pointer of the base class to point to an object of the derived class and then calling the function from that pointer.

Q6. How do you invoke a base member function from a derived class in which you have not overridden that function?

Ans. By using the pointer of the base class to point to an object of the derived class and calling the function through the pointer.

Q7. If a base class declares a function to be virtual, and a derived class does not use the term virtual when overriding that class, is it still virtual when inherited by a third-generation class?

Ans. The virtual keyword can be used when declaring overriding functions in a derived class, but it is unnecessary; overrides of virtual functions are always virtual. Virtual functions in a base class must be defined unless they are declared using the pure-specifier.

Q8. What is the protected keyword used for?

Ans. The protected keyword is an access modifier used for attributes, methods and constructors, making them accessible in the same package and subclasses.