OOP LABORATORY 6

Name: **ANIRBAN HAZRA**
Section: **B-12**
Roll : **2005643**

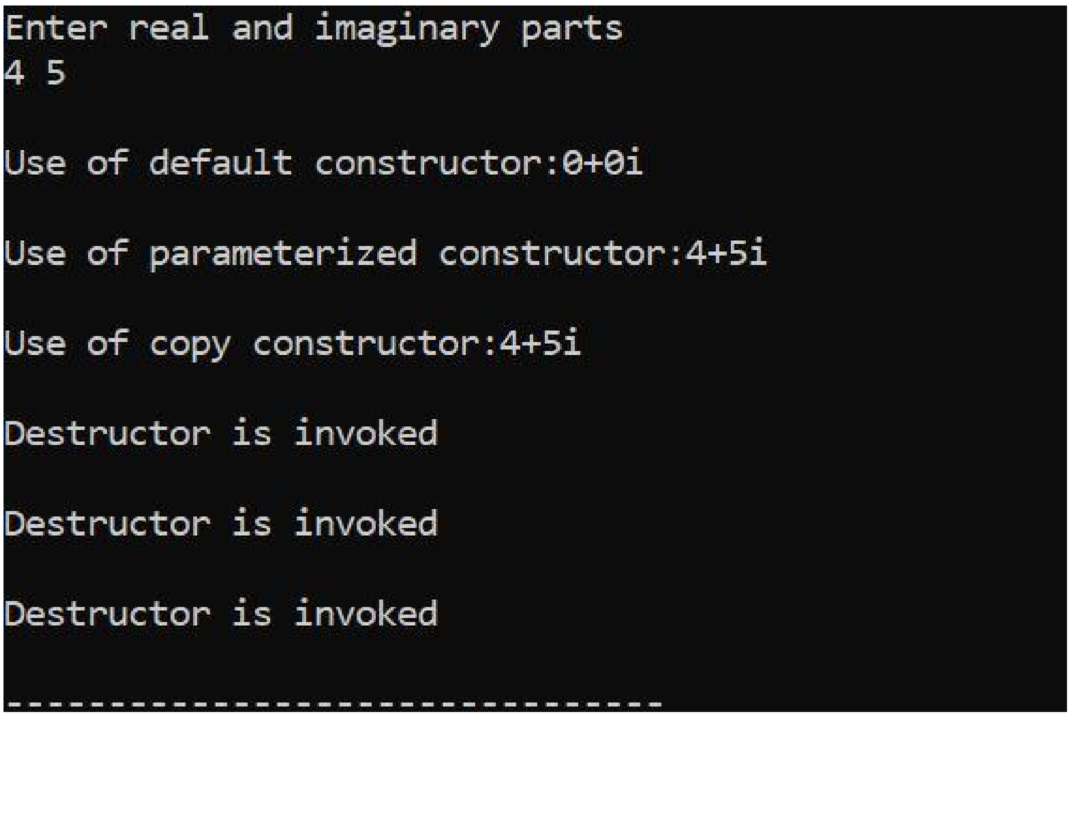
1. Create a class complex which stores real and imaginary part of a complex number. Include all types of constructors and destructor. The destructor should display a message about the destructor being invoked. Create objects using different constructors and display them.

PROGRAM CODE:-

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

class complex
{
   int real , img;

public:

   complex()
   {
      real=0;
      img=0;
   }

   complex(int r, int i)
   {
      real=r;
      img=i;
   }

   complex(const complex &c1)
   {
      real = c1.real;
      img = c1.img;
   }

   void disp()
   {
      cout<<real<<"+"<<img<<"i"<<endl;
   }
```

```
    ~complex()
    {cout<<"\nDestructor is invoked\n";}

};

int main()
{
    int x,y;
    complex c1;

    cout<<"Enter real and imaginary parts\n";
    cin>>x>>y;

    complex c2(x,y);
    complex c3=c2;

    cout<<"\nUse of default constructor:";
    c1.disp();

    cout<<"\nUse of parameterized constructor:" ;
    c2.disp();

    cout<<"\nUse of copy constructor:" ;
    c3.disp();

    return 0;
}
```

<u>OUTPUT:-</u>



```
Enter real and imaginary parts
4 5

Use of default constructor:0+0i

Use of parameterized constructor:4+5i

Use of copy constructor:4+5i

Destructor is invoked

Destructor is invoked

Destructor is invoked


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

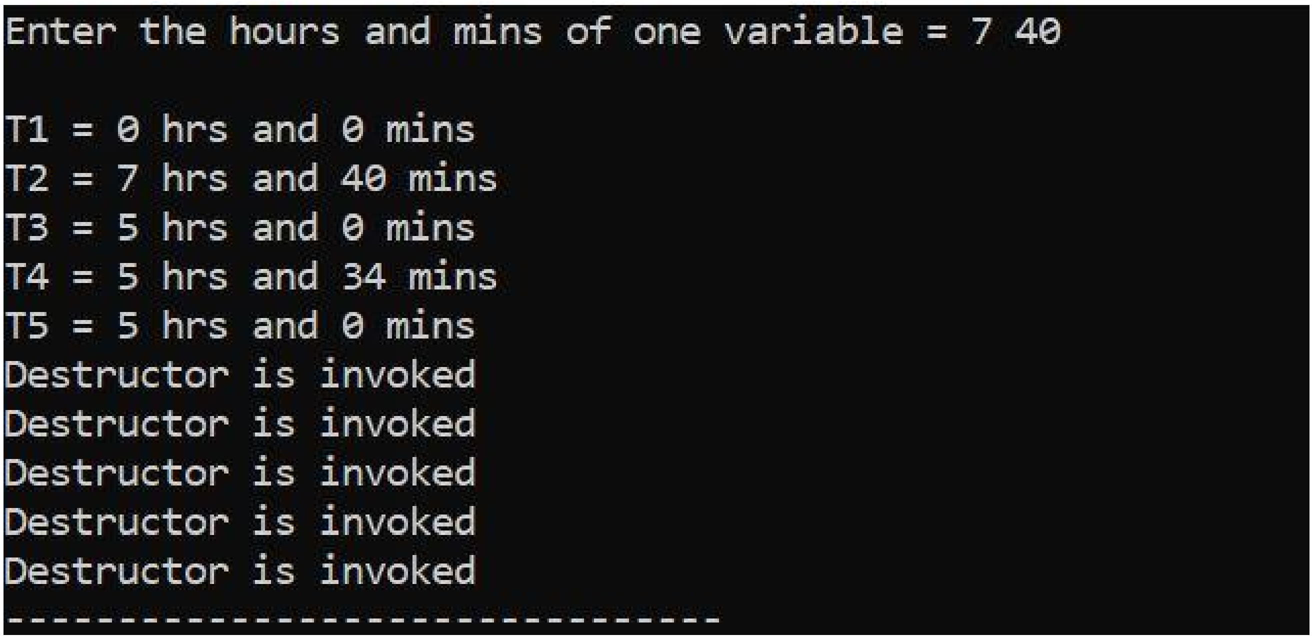
2. Create a class which stores time in hh:mm format. Include all the constructors. The parameterized constructor should initialize the minute value to zero, if it is not provided.

PROGRAM CODE:-

```
#include<iostream>

using namespace std;

class timer
{
        int hh;
        int mm;
        public:

        timer()
        {
                hh=0;
                mm=0;
        }

        timer(int a,int b=0)
        {
                hh=a;
                mm=b;
        }

        timer(timer &t)
        {
                hh=t.hh;
                mm=t.mm;
        }

        void input()
        {
                cin>>hh>>mm;
        }

        void display()
        {
                cout<<hh<<" hrs and "<<mm<<" mins";
        }

        ~timer()
        {
                cout<<"\nDestructor is invoked";
        }
};
```

```
int main()
{
        timer t1,t2;
        timer t3(5);
        timer t4(5,34);
        timer t5=t3;

        cout<<"Enter the hours and mins of one variable = ";
        t2.input();

        cout<<"\nT1 = ";
        t1.display();

        cout<<"\nT2 = ";
        t2.display();

        cout<<"\nT3 = ";
        t3.display();

        cout<<"\nT4 = ";
        t4.display();

        cout<<"\nT5 = ";
        t5.display();

        return 0;
}
```

OUTPUT:-



```
Enter the hours and mins of one variable = 7 40

T1 = 0 hrs and 0 mins
T2 = 7 hrs and 40 mins
T3 = 5 hrs and 0 mins
T4 = 5 hrs and 34 mins
T5 = 5 hrs and 0 mins
Destructor is invoked
Destructor is invoked
Destructor is invoked
Destructor is invoked
Destructor is invoked
--------------------------------
```

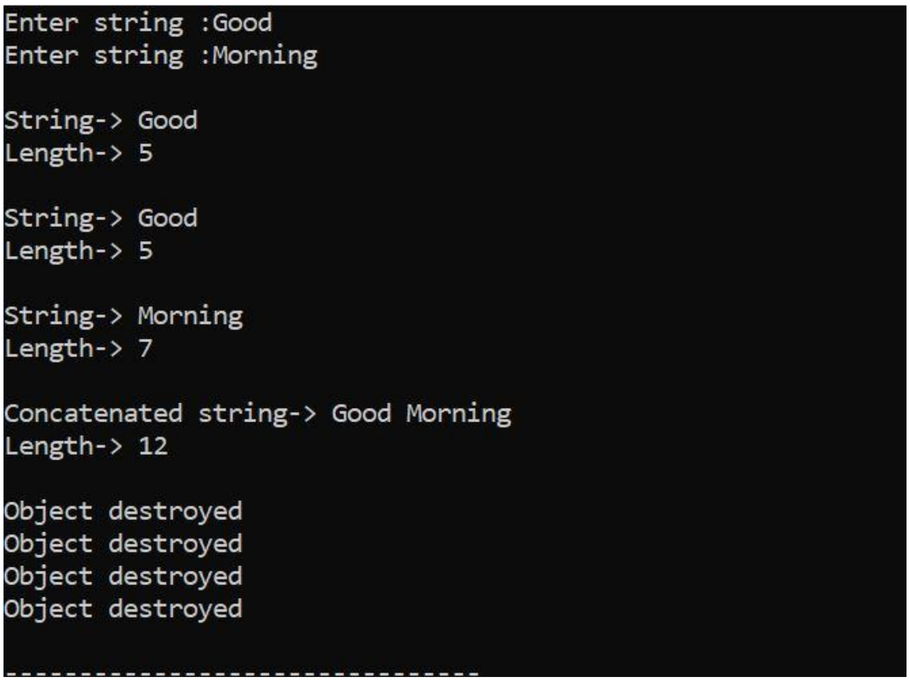
3. Create a class which stores a sting and its length as data members. Include all the constructors. Include a member function to join two strings and display the concatenated string.

PROGRAM CODE:-

```
#include<iostream>
#include<stdlib.h>
#include<string.h>
using namespace std;

class con
{
   private:
   char *s=new char[300];
   int l;

   public:
   con()
   {
      s=new char[300];
      l=0;
   }
   con(char *str, int len)
   {
      s=str;
      l=len;
   }
   con(con &f)
   {
      s=f.s;
      l=f.l;
   }
   ~con()
   {
      cout<<"Object destroyed\n";
   }

   public:
   void input()
   {
      cout<<"Enter string :";
      cin.getline(s, 300);
      l=strlen(s);
      cout<<endl;
   }
   void join(con &f, con &t)
   {
      strcat(f.s, t.s);
      cout<<"Concatenated string-> "<<f.s<<endl;
```

```
        cout<<"Length-> "<<strlen(f.s)<<endl;
        cout<<endl;
    }
    void display()
    {
        cout<<"String-> "<<s<<endl;
        cout<<"Length-> "<<l<<endl;
        cout<<endl;
    }
};

int main()
{
    con o1;
    char *str=new char[300];
    cout<<"Enter string :";
    cin.getline(str, 300);
    int len=strlen(str);
    con o2(str, len);
    con o3=o2;
    con o4;
    o4.input();
    o2.display();
    o3.display();
    o4.display();
    o2.join(o3, o4);

    return 0;
}
```

OUTPUT:-



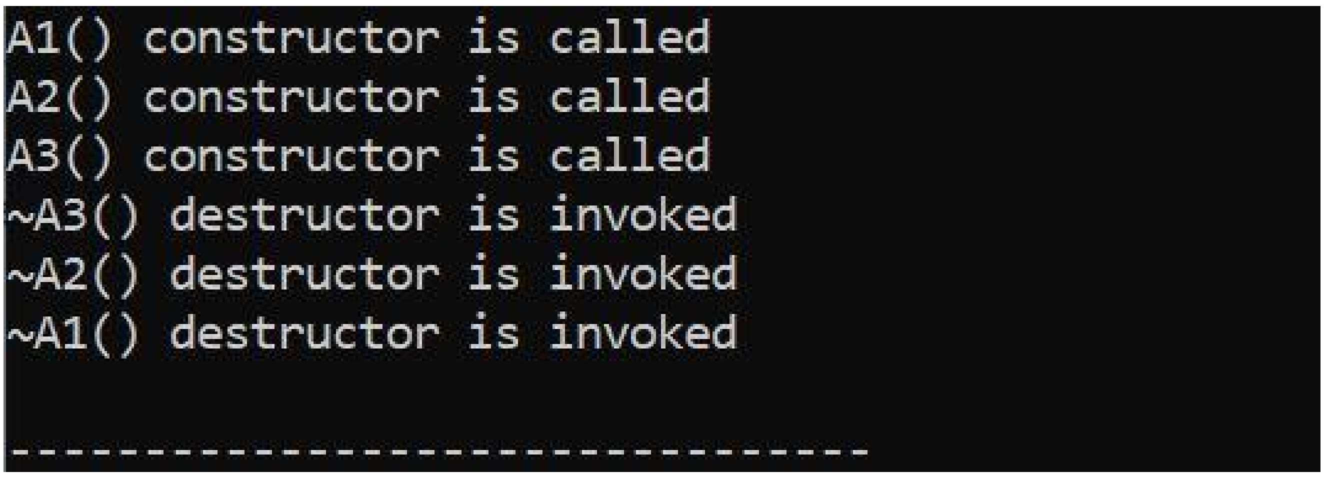
```
Enter string :Good
Enter string :Morning

String-> Good
Length-> 5

String-> Good
Length-> 5

String-> Morning
Length-> 7

Concatenated string-> Good Morning
Length-> 12

Object destroyed
Object destroyed
Object destroyed
Object destroyed

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

4. WAP to demonstrate the order of call of constructors and destructors for a class.

PROGRAM CODE:-

```
#include <iostream>
using namespace std;
class A
{
    int i;

    public:

        A(int a = 0)
        {
            i=a;
            cout << "A"<<i<<"() constructor is called "<<endl;
        }

        ~A()
        {
            cout << "~A"<<i<<"() destructor is invoked "<<endl;
        }
};

int main()
{

    A a1(1);
    A a2(2);
    A a3(3);

    return 0;
}
```

OUTPUT:-



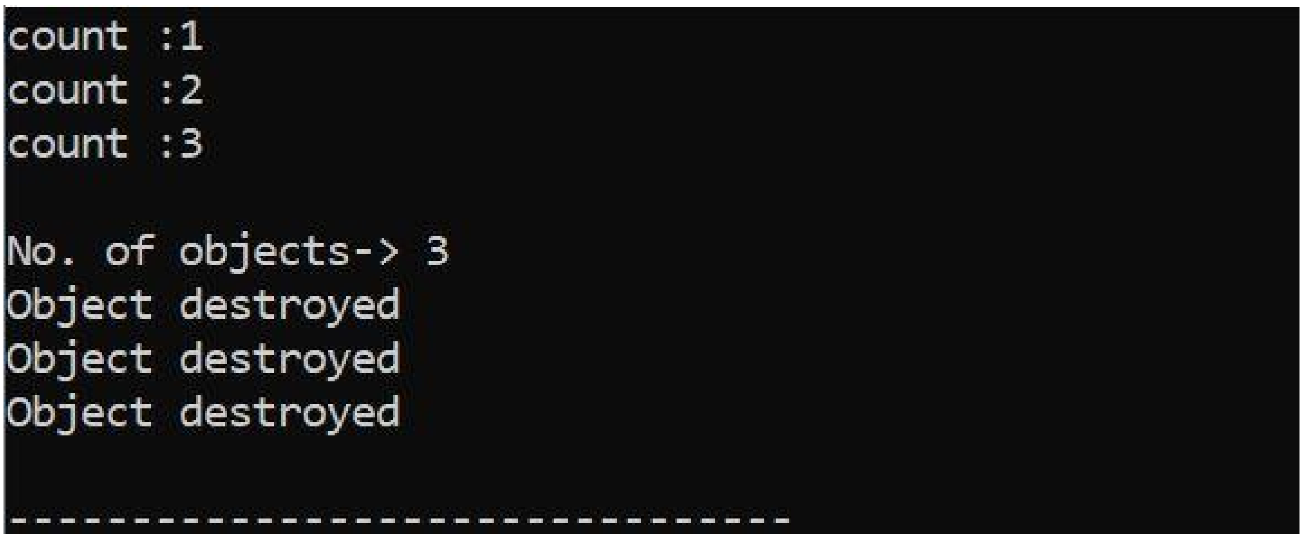
```
A1() constructor is called
A2() constructor is called
A3() constructor is called
~A3() destructor is invoked
~A2() destructor is invoked
~A1() destructor is invoked


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

5. WAP to count number of objects created from a class using concept of static data members and static member function.

PROGRAM CODE:-

```
#include<iostream>
#include<stdlib.h>

using namespace std;

class test
{
   private:

   int n;
   static int c;

   public:

   test()
   {

          n=0;
   }

   ~test()
   {

         cout<<"Object destroyed\n";
   }

   void count()
   {

         c++;
         cout<<"count :"<<c<<endl;
   }

   static void display(void)
   {

         cout<<endl;
         cout<<"No. of objects-> "<<c<<endl;
   }

};


int test::c=0;
int main()
{

   test o1,o2,o3;

   o1.count();
   o2.count();
   o3.count();

   test::display();

   return 0;

}
```

<u>OUTPUT:-</u>



```
count :1
count :2
count :3

No. of objects-> 3
Object destroyed
Object destroyed
Object destroyed


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