Object Oriented Programming

Lab Report

Lab14

Group Members Name & Reg #:	**Muhammad Haris Irfan** **(FA18-BCE-090)**
Class	Object Oriented Programming CSC241 (**BCE-4B**)
Instructor's Name	Maam Amber Madeeha Zeb

In Lab Tasks

5.1 Question 1:

Practice the above mentioned examples

Solution:

The Above-mentioned codes were practiced.

5.2 Question 2:

Create a class which only works for absolute numbers, if it encounters any negative occurrence, then it throw an exception to its handler and display errors.

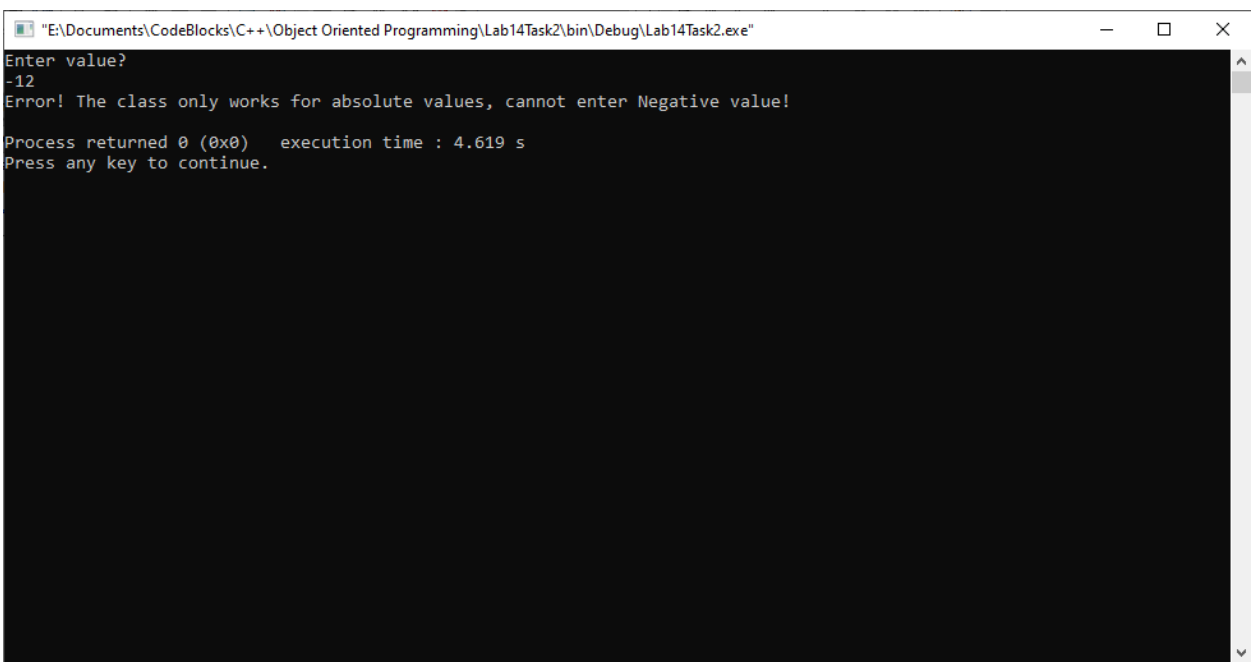
Solution:

The code is given below,

```
#include <iostream>

using namespace std;

class absolute
{
private:

    int t=0;

public:



     void getvalue()
     {
```

```
    try
    {
        cout<<"Enter value?"<<endl;
        cin>>t;

        if(t<0)
        throw (t);

    }

     catch(int )
    {
        cout<<"Error! The class only works for absolute values, cannot enter Negative value!"<<endl;
    }
     }

};

int main()
{


     absolute a1;
     a1.getvalue();


    return 0;
}
```

Console Output is shown below.



5.3 Question 3:

Modify the above task, by creating an exception class with an error code and corresponding error message. Code and message should be thrown and displayed in catch block.

Solution:

The code is given below,

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


 class exceptionn
 {
         private:
             int t=1;
         public:

         void func()
         {
           try
           {
              if(t!=0)
               throw t;
           }

           catch(int)
           {
               cout<<"Error!"<<endl;
               throw t;
           }

         }

         };
 int main()
 {

 exceptionn e1;
 try
 {
 e1.func();
 }

 catch(int)
           {
               cout<<endl<<"Value should not be less then 0!"<<endl;
           }

     return 0;
 }
```

Console Output is shown below.

```
"E:\Documents\CodeBlocks\C++\Object Oriented Programming\Lab14Task3\bin\Debug\Lab14Task3.exe"
Error!

Value should not be less then 0!

Process returned 0 (0x0)   execution time : 0.015 s
Press any key to continue.
```

POST LAB

6.1 Question 4:

The *queue* is another data structure. A physical analogy for a queue is a line at a bank. When you go to the bank, customers go to the *rear* (end) of the line and customers who are serviced come out of the line from the *front* of the line.

The main property of a queue is that objects go on the *rear* and come off of the *front* of the queue.

- *Make-Queue*
  Create a new, empty queue object.
- *Empty*
  Reports whether queue is empty or not.
- *Enter(or Insert)*
  Places an object at the rear of the queue
- *Delete (or Remove)*
  Removes an object from the *front* of the queue and produces that object.

Write a program to create a queue class and do queue operations with exception handling.

Solution:

I am attaching my code below,

```
#include <iostream>

using namespace std;


class que
{
private:
   int que[5];
   int index=5;
   int frontqu=-1;
   int rear=-1;
public:

    void chckempty()
    {
        try
        {


```

```
        if(frontqu==-1)
            throw frontqu;
        else
            cout<<"Queue is not Empty"<<endl;

        }

        catch (int)
        {
            cout<<"Queue is Empty"<<endl<<endl;
        }

    }

    void enqueue()
    {

   int val;
   try
   {

   if (rear == index - 1)
        throw rear;
   else
    {
      if (frontqu == - 1)
      frontqu = 0;
      cout<<"Insert the element in queue : "<<endl;
      cin>>val;
      cout<<"Element Inserted!: "<<endl;
      rear++;
      que[rear] = val;
   }

   }

   catch (int)
   {
       cout<<endl<<"Queue is Full"<<endl<<endl;
   }

    }


    void dequeue()
    {

    try{

   if (frontqu == - 1 || frontqu > rear)
    {

      throw(frontqu) ;
   }


   else {
      cout<<"Element deleted from queue is : "<< que[frontqu] <<endl;
      frontqu++;;
   }
    }
     catch(int)
    {

    cout<<endl<<"Queue Underflow! The Queue is Empty! "<<endl;

    }
    }

};

```

```
int main()
{
    que q1;

    q1.chckempty();

    q1.enqueue();
    q1.enqueue();
    q1.enqueue();
    q1.enqueue();
    q1.enqueue();
    q1.enqueue();

    q1.dequeue();
    q1.dequeue();
    q1.dequeue();
    q1.dequeue();
    q1.dequeue();
    q1.dequeue();




    return 0;
}
```

The result for this program is shown below,

```
"E:\Documents\CodeBlocks\C++\Object Oriented Programming\LAb14Task4\bin\Debug\LAb14Task4.exe"
Queue is Empty

Insert the element in queue :
1
Element Inserted!:
Insert the element in queue :
2
Element Inserted!:
Insert the element in queue :
3
Element Inserted!:
Insert the element in queue :
4
Element Inserted!:
Insert the element in queue :
5
Element Inserted!:

Queue is Full

Element deleted from queue is : 1
Element deleted from queue is : 2
Element deleted from queue is : 3
Element deleted from queue is : 4
Element deleted from queue is : 5

Queue Underflow! The Queue is Empty!

Process returned 0 (0x0)   execution time : 3.933 s
Press any key to continue.
```

_____THE END_____