

# Object Oriented Programming

## Lab Report

### Lab14



Group Members Name & Reg #:	<b><u>Muhammad Haris Irfan</u></b> <b>(FA18-BCE-090)</b>
Class	Object Oriented Programming CSC241 ( <b>BCE-4B</b> )
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.

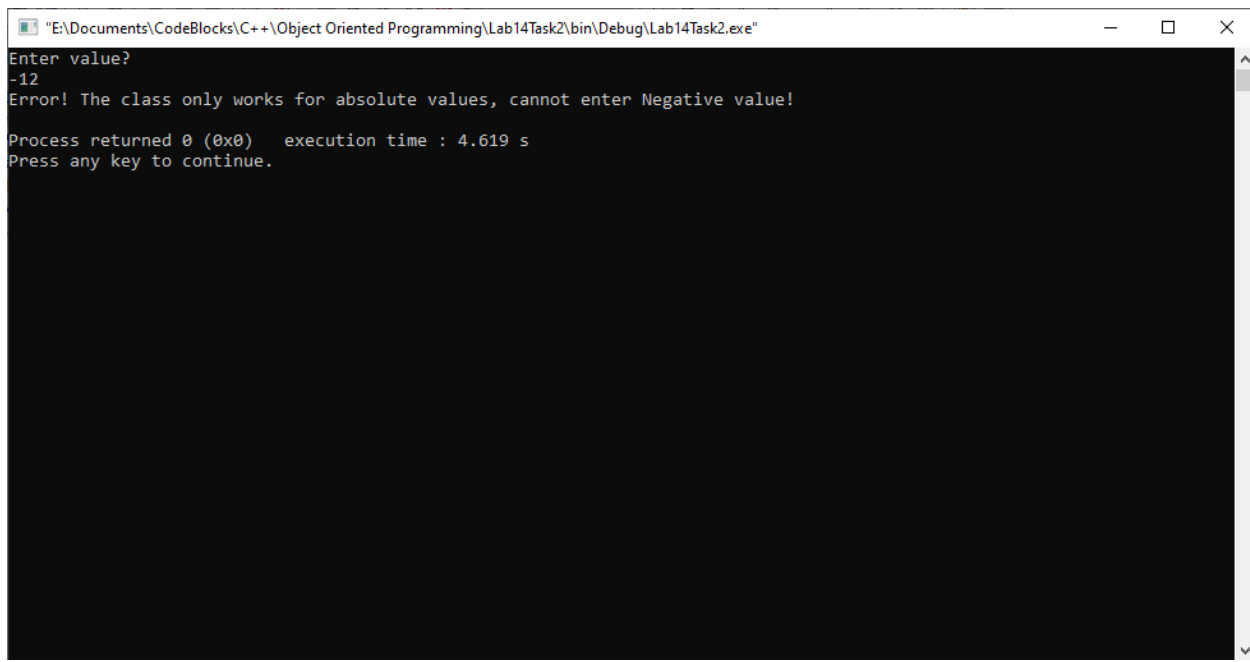
### Solution:

The code is given below,

```
1  #include <iostream>
2
3  using namespace std;
4
5  class absolute
6  {
7  private:
8
9      int t=0;
10
11 public:
12
13
14
15     void getvalue()
16     {
```

```
17     try
18     {
19         cout<<"Enter value?"<<endl;
20         cin>>t;
21
22         if(t<0)
23             throw (t);
24
25     }
26
27     catch(int )
28     {
29         cout<<"Error! The class only works for absolute values, cannot enter Negative
value!"<<endl;
30     }
31 }
32
33 };
34
35 int main()
36 {
37
38     absolute a1;
39     a1.getvalue();
40
41
42     return 0;
43 }
44 }
```

**Console Output is shown below.**



```
"E:\Documents\CodeBlocks\C++\Object Oriented Programming\Lab14Task2\bin\Debug\Lab14Task2.exe"
Enter value?
-12
Error! The class only works for absolute values, cannot enter Negative value!

Process returned 0 (0x0)   execution time : 4.619 s
Press any key to continue.
```

## 5.3 Question 3:

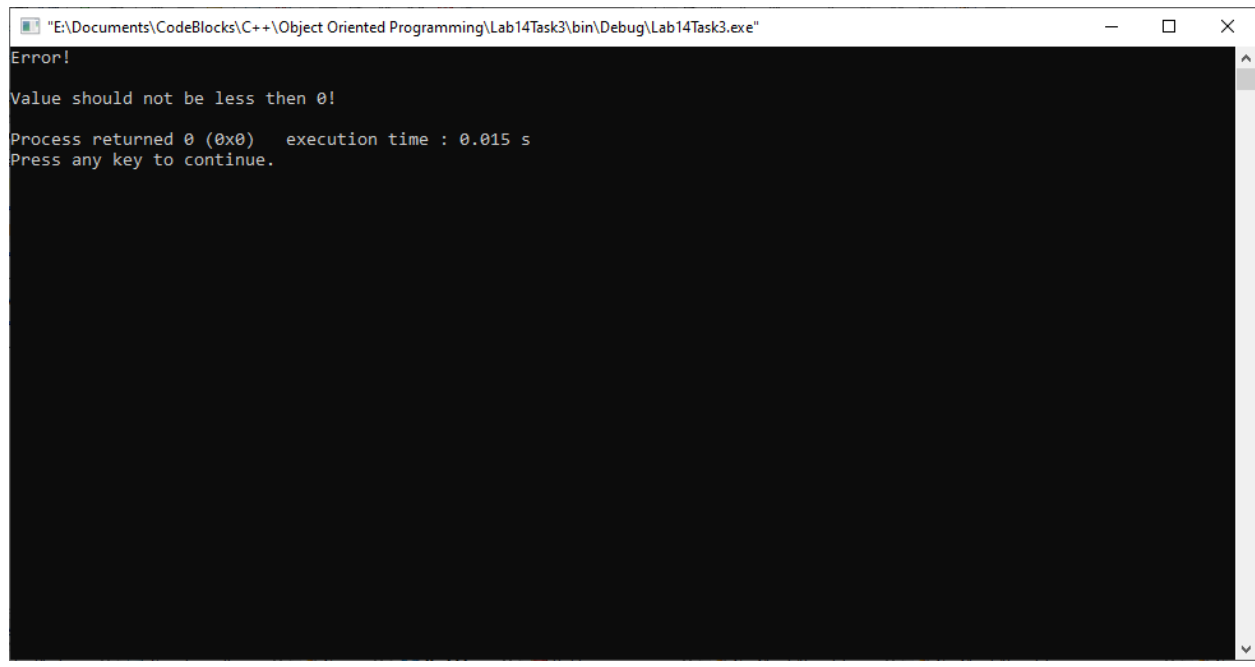
Modify the above task, by creating an exception class with an error code and corresponding error message. Code and message should be thrown and displayed in catch block.

## Solution:

The code is given below,

```
1  #include <iostream>
2  #include<conio.h>
3  using namespace std;
4
5
6  class exceptionn
7  {
8      private:
9          int t=1;
10     public:
11
12     void func ()
13     {
14         try
15         {
16             if(t!=0)
17                 throw t;
18         }
19
20         catch(int)
21         {
22             cout<<"Error!"<<endl;
23             throw t;
24         }
25     }
26
27 };
28
29 int main()
30 {
31
32     exceptionn e1;
33     try
34     {
35         e1.func ();
36     }
37
38     catch(int)
39     {
40         cout<<endl<<"Value should not be less than 0!"<<endl;
41     }
42
43     return 0;
44 }
```

## Console Output is shown below.



The screenshot shows a Windows console window titled "E:\Documents\CodeBlocks\C++\Object Oriented Programming\Lab14Task3\bin\Debug\Lab14Task3.exe". The console output is as follows:

```
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,

```
1  #include <iostream>
2
3  using namespace std;
4
5
6  class que
7  {
8  private:
9      int que[5];
10     int index=5;
11     int frontqu=-1;
12     int rear=-1;
13 public:
14
15     void chkempty ()
16     {
17         try
18         {
19
20
```

```

21         if(frontqu==-1)
22             throw frontqu;
23         else
24             cout<<"Queue is not Empty"<<endl;
25     }
26 }
27
28 catch (int)
29 {
30     cout<<"Queue is Empty"<<endl<<endl;
31 }
32
33 }
34
35 void enqueue ()
36 {
37
38     int val;
39     try
40     {
41
42         if (rear == index - 1)
43             throw rear;
44         else
45         {
46             if (frontqu == - 1)
47                 frontqu = 0;
48             cout<<"Insert the element in queue : "<<endl;
49             cin>>val;
50             cout<<"Element Inserted!: "<<endl;
51             rear++;
52             que[rear] = val;
53         }
54     }
55 }
56
57 catch (int)
58 {
59     cout<<endl<<"Queue is Full"<<endl<<endl;
60 }
61
62 }
63
64
65 void dequeue ()
66 {
67
68     try{
69
70     if (frontqu == - 1 || frontqu > rear)
71     {
72
73         throw (frontqu) ;
74     }
75
76     else {
77         cout<<"Element deleted from queue is : "<< que[frontqu] <<endl;
78         frontqu++;
79     }
80 }
81 catch(int)
82 {
83
84
85     cout<<endl<<"Queue Underflow! The Queue is Empty! "<<endl;
86
87 }
88 }
89
90 };
91

```

```

92  int main ()
93  {
94      que q1;
95
96      q1.chkempty ();
97
98      q1.enqueue ();
99      q1.enqueue ();
100     q1.enqueue ();
101     q1.enqueue ();
102     q1.enqueue ();
103     q1.enqueue ();
104
105     q1.dequeue ();
106     q1.dequeue ();
107     q1.dequeue ();
108     q1.dequeue ();
109     q1.dequeue ();
110     q1.dequeue ();
111
112
113
114
115
116     return 0;
117 }

```

**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\_\_\_\_\_

---