창의적 소프트웨어 프로그래밍 Lab 19

Handed out : Thu, Dec 01, 2022

Do not need to submit answer!

You can ask a question by e-mail.

1. Write a program that works as follows:

   A. Complete exception handling in the following code.

      i. If a bad allocation occurs (when n≤0) in the constructor, throw an exception and catch it in the main() and prints "caught in the main".

      ii. Objects created inside the try block calls destructors when an exception occurs, so make sure you implement the destructor properly, and check which constructors and destructors are called.

B.

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

class A
{
	public:
		A(int n)
		{
			//implement something here
			cout << "ID=" << n << ": constructed\n";

			n_ID = n;
			data = new int[n];
		}
		~A()
		{
			cout << "ID=" << n_ID << ": destroyed\n";
			//implement something here
		}
	private:
		int* data = NULL;
		int n_ID;
};

int main(){
	try{
		A a(3);
		A b(2);
		{
			A c(1);
			A d(0);
			A e(-1);
		}
	}
	//implement something here
	return 0;
}
```

C. Input: None

D. Output: Printed results as follows.

```
$ ./exception1
ID=3: constructed
ID=2: constructed
ID=1: constructed
ID=1: destroyed
ID=2: destroyed
ID=3: destroyed
caught in the main
$
```

2. Write a program that works as follows:
   A. If you throw an object 'a' as shown in the following code, 'a' is copied and used internally because the destructor of 'a' is called at the end of the try scope (ie, “Objects Thrown as Exceptions Are Always Copied ”).
   B. When you run the following program, modify the program so that it prints out as shown in the example output.
   C. Hint: What do you need for an object to be copied properly?

D.

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

int data_size = 5;
class A{
       public:
              A(){
                     data = new int[data_size];
                     for(int i = 0; i < data_size; i++)
                            data[i] = i;
                     cout << "constructed\n";
              }
              ~A()
              {
                     for(int i = 0; i < data_size; i++)
                            data[i] = 0;
                     delete[] data;
                     data = NULL;
                     cout << "destroyed\n";
              }

       private:
              int* data = NULL;

       friend ostream& operator <<(std::ostream& os, const A& a);
};

ostream& operator <<(std::ostream& os, const A& a)
{
    for(int i=0; i<data_size; i++)
       os << a.data[i] << " ";
    return os;
}

int main()
{
       try
        {
              A a;
              cout << a << endl;
              throw a;
       }
       catch(A& a)
       {
              cout << "err. handled\n";
                     cout << a << endl;
       }
       return 0;
}
```

E. Input: None

F. Output: Printed results as follows.

```
$ ./exception2
constructed
0 1 2 3 4
destroyed
err. handled
0 1 2 3 4
destroyed
$
```