05/09/2024

Object-Oriented Programming – Second Midterm

(Open textbook, notes, compiler, 180 minutes)

Total Score **100%**

Name: _____

ID: _____

Grading rule

I. Your program must be **compiled without errors to earn credits for that problem.**

II. The corresponding outputs should be produced, even in terms of **format**.

III. If you cannot achieve the example output, you can write comments in your code to explain reasons why, and only then partial credits could be given.

1. **(25%)** Write a `Rectangle` class, which has private data members: `string color`, `double length` and `double width`.

   Please separate the declaration and the definition of the Rectangle class in Rectangle.h and Rectangle.cpp.

   If you declare a friend function **inside** of the Rectangle class, please define the friend function in Rectangle.h but **outside** of the class definition of Rectangle.

   Please add required member functions.

   You CANNOT change the following file problem1_main.cpp.

```
#include <iostream>
#include "Rectangle.h"

using namespace std;

int main(){
  cout << "Start of main() "<< endl;
  cout << "1. ";
  Rectangle r1("Yellow",30, 40);
  cout << r1 << endl;

  cout << "2. ";
  Rectangle r2(r1);
  cout << r2 << endl;
  cout << "3. is r2 square? " << r2.isSquare() << endl;

  cout << "4. ";
  Rectangle r3;
  cout << r3 << endl;

  cout << "5. ";
  r2.changeColor("Green");
  r2.changeWidth(100);
```

```
  r2.changeLength(100);
  r3 = r2;
  cout << r3 << endl;

  cout << "6. is r3 square? " << r3.isSquare() << endl;

  Rectangle r4(2,3);
  cout << "7. " << r4 << endl;
  cout << "End of main() "<< endl;
}
```

Output:

```
Start of main()
1. Color: Yellow, Length: 30, Width: 40
2. Color: Yellow, Length: 30, Width: 40
3. is r2 square? 0
4. Color: White, Length: 1, Width: 1
5. Color: Green, Length: 100, Width: 100
6. is r3 square? 1
7. Color: White, Length: 2, Width: 3
End of main()
```

2. **(25%)** Please complete the `MyVector` class, which internally uses a pointer to a **raw double array** with a member of **int size**. Please complete required member functions. You need to properly manage the memory. Producing the output does not mean you receive full credits if memory is not managed correctly.

The following is the main function you **CANNOT change**.

```
int main(){
  MyVector vec(5);
  vec[0] = 0.2;
  vec[1] = 3.1;
  vec.printMyVector();

  vec.resize(8);
  vec[6] = 0.4;
  vec[7] = 3.9;
  vec.printMyVector();

  MyVector vec2;
  vec2.printMyVector();
  vec2.resize(2);
  vec2[1] = 3;
  vec2.printMyVector();
}
```

Output:

```
printing MyVector: 0.2 3.1 0 0 0
printing MyVector: 0.2 3.1 0 0 0 0 0.4 3.9
MyVector has 0 size
printing MyVector: 0 3
cleaning up MyVector of size 2
cleaning up MyVector of size 8
```

3. **(25%)** In this problem, you will implement a class called `SharedArray`. It has private members: `int size` and `int* data`. For any object created from this class, they all share the same copy (memory) of `data`. This class has a friend function for the "<<" operator. You should add required members such as data, constructors, operators and functions.

The following are functions you cannot change:

```
SharedArray create(){
     return SharedArray(5);
}

int main(){
    SharedArray m;
    cout << "before call to create()" << endl;
    m = create();
    m[0] = 5;
    cout << "m: " << m;
    const SharedArray n(m);
    m[0] = 1;
    m[2] = n[0];
    cout << "m: " << m;
    cout << "n: " << n;
    SharedArray o;
    o = m;
    cout << "o: " << o;
    SharedArray p = move(create());
    cout << "before returning from main" << endl;
    return 0;
}
```

Output:

```
Default Constructor is called
before call to create()
Constructor is called
Move Assignment
Destructor is called but data still in use by other object!
instances left: 1
m: 5 0 0 0 0
Copy Constructor is called -Shallow copy
m: 1 0 1 0 0
n: 1 0 1 0 0
Default Constructor is called
Copy Assignment
o: 1 0 1 0 0
Constructor is called
Move Constructor
Destructor is called but data still in use by other object!
instances left: 4
before returning from main
Destructor is called but data still in use by other object!
instances left: 3
Destructor is called but data still in use by other object!
instances left: 2
Destructor is called but data still in use by other object!
instances left: 1
Destructor is called and clean up is done!
instances left: 0
```

4. **(25%)** In this problem, you are implementing two classes: Student and Class. One student only enrolls in a class, and a class can have many students.

   The Student class has a **smart pointer** to the class it is enrolled in. The Class class has a **vector** of **smart pointer**s to students in its class. The following is the main you **cannot** change:

```
int main() {
  shared_ptr<Class> mathClass = make_shared<Class>("Math");
  shared_ptr<Class> englishClass = make_shared<Class>("English");

  shared_ptr<Student> alice = make_shared<Student>("Alice");
  shared_ptr<Student> bob = make_shared<Student>("Bob");
  shared_ptr<Student> charlie = make_shared<Student>("Charlie");
  shared_ptr<Student> julian = make_shared<Student>("Julian");

  mathClass->registerStudent(alice);
  alice->setClass(mathClass);

  mathClass->registerStudent(bob);
  bob->setClass(mathClass);

  englishClass->registerStudent(julian);
  julian->setClass(englishClass);

  englishClass->registerStudent(charlie);
  charlie->setClass(englishClass);

  cout << alice->getName() << " is in: " << alice->getClassName() << endl;
  cout << bob->getName() << " is in: " << bob->getClassName() << endl;
  cout << charlie->getName() << " is in: " << charlie->getClassName() <<
endl;
  cout << julian->getName() << " is in: " << julian->getClassName() << endl;

  mathClass->printStudents();
  englishClass->printStudents();

  return 0;
}
```

Output:

```
Alice is in: Math
Bob is in: Math
Charlie is in: English
Julian is in: English
Students in Math:
Alice
Bob
Students in English:
Julian
Charlie
Class destructor: English
Student destructor: Julian
Student destructor: Charlie
Class destructor: Math
Student destructor: Alice
Student destructor: Bob
```