LASALLE COLLEGE VANCOUVER

Smart Pointers

VGP 131 - Object Oriented Programming in C++ II

Instructor: Ivaldo Tributino

May 11, 2022

ASSIGNMENT INSTRUCTION

- The exam must be submitted by May 15, 2022.
- Each problem presents its own score, the sum of all scores is 100.

Student's Number:
Student's Name:

(10 POINTS) PROBLEM 1

Follow the sequence below to create a function to resize arrays.

1. create a new array,
2. copy the contents of the old array to the new array,
3. delete the old array,
4. make your pointer variable point to the new array.

To dynamically resize an array.

```
void arrayResize(int*& A, int oldsize, int newsize){
   //new array

   //copy old array to new array

   //delete old array

   //point old array to new array
}
```

(10 POINTS) PROBLEM 2

Write a program that defines a base class with a pure virtual member function. Create a derived class that overrides a virtual function in the base class. Create a polymorphic object of a derived class through a unique pointer to a base class. Finally, Invoke the overridden member function through a unique pointer.

(10 POINTS) PROBLEM 3

Is there something wrong with the following codes such as a memory leak or a dangling pointer? Explain your answer.

A

```
unique_ptr<Polygon> unPolyPtr = make_unique<Polygon>(2,10);
Polygon *polyPtr;
polyPtr = unPolyPtr.release();
```

B

```
unique_ptr<Polygon> unPolyPtr = make_unique<Polygon>(2,10);
Polygon *polyPtr;
polyPtr = unPolyPtr.get();
unPolyPtr.reset(nullptr);
```

C

```
unique_ptr<Polygon> unPolyPtr = make_unique<Polygon>(2,10);
Polygon *polyPtr;
polyPtr = unPolyPtr.release();
unPolyPtr.get_deleter()(polyPtr);
```

D

```
shared_ptr<Polygon> shPtr = make_shared<Polygon>(3,4);
Polygon* rowPtr = shPtr.get();
delete rowPtr;
```

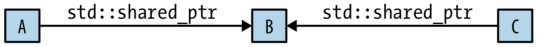
(10 POINTS) PROBLEM 4

What is the problem with the code below?

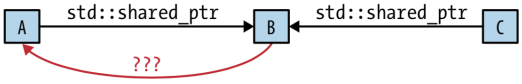
```
Polygon* poly = new Polygon(1,6);

shared_ptr<Polygon> shPtrPoly(poly);
cout << shPtrPoly.use_count() << endl;

shared_ptr<Polygon> shPtrPoly2(poly);
cout << shPtrPoly2.use_count() << endl;
```

(10 POINTS) PROBLEM 5

Consider a data structure with objects A, B, and C in it, where A and C share ownership of B and therefore hold std::shared_ptrs to it:



Suppose it'd be useful to also have a pointer from B back to A. What kind of pointer should this be? Explain your answer.



(10 POINTS) PROBLEM 6

Complete the code below:

```
vector<Polygon*> polyMatrix;

polyMatrix.resize(5);

for(int i = 0; i < polyMatrix.size(); ++i){
    polyMatrix[i] = new Polygon[3]{Polygon(1,i+3),Polygon(2,i+3),Polygon(3,
    i+3)};
}

for(int i = 0; i< polyMatrix.size(); ++i){

}
```

to get the following output (constructor and destructor invoked 15 times).

```
...
Constructor Invoked
Constructor Invoked
0.433013 | 1.73205 | 3.89711 |
1 | 4 | 9 |
1.72048 | 6.88191 | 15.4843 |
2.59808 | 10.3923 | 23.3827 |
3.63391 | 14.5356 | 32.7052 |
Polygon destroyed
Polygon destroye
...
```

(10 POINTS) PROBLEM 7

Now let's create a program similar to the previous problem using smart pointer (Complete the code below).

```
vector<shared_ptr<Polygon>> smartPolyMatrix;
smartPolyMatrix.resize(5);

for(int i = 0; i < smartPolyMatrix.size(); ++i){

}

for(int i = 0; i< smartPolyMatrix.size(); ++i){
    for(int j = 0; j < 3; ++j){

    }
}
```

(15 POINTS) PROBLEM 8

Now create a two-dimensional array (multidimensional dynamic arrays) from scratch. Where it contains the same elements as the previous matrix.

```
Polygon ** array2D = new Polygon*[5];
```

(15 POINTS) PROBLEM 9

Implementing a singly linked list with smart pointers.