**Object-Oriented Programming**

**10/16/2018**

# Lab Assignment 6

**Lab Grading Policy: Attendance 40%, Score 60%**

In case you have difficulty in finishing the exercises on time, you should upload them by **Thursday noon** with a penalty of 20% on your score. No late submission is permitted after that. We will in general post the reference solutions **by Friday**.

---

1. **(20%)** Please using the following declarations to complete this problem.

```
struct IntArray{
    int* ia;
    int n;
};

IntArray creatIntArray();
int findMax(const IntArray&);
void printIntArray(const IntArray&);
void deleteIntArray(IntArray&);
```

In this problem, the `IntArray` structure is used as the data to be passed around functions. The first function requires the dynamic allocation of an array of integers for the `ia` member of the `IntArray` structure. You will use the `new` keyword for the memory allocation. To prevent memory leak, you will deallocate the array in the function `deleteIntArray(IntArray&)`, using the `delete` keyword. [You will have a HW problem to investigate on memory leak.]

The following is the main method you cannot change.

```
int main(){
    IntArray intArray = creatIntArray();
    int i = findMax(intArray);
    cout << "Max value in integer array is: " << intArray.ia[i] <<
endl;
    printIntArray(intArray);
    deleteIntArray(intArray);

    return 0;
}
```

Example runs:

```
How many integers do you want to input: 5
Please input the integers: 1 2 3 4 5
Max value in integer array is: 5
Integer Array: 1 2 3 4 5
```

```
How many integers do you want to input: 3
Please input the integers: 1 3 2
Max value in integer array is: 3
Integer Array: 1 3 2
```

2. **(20%)** Create a class `RandomArray` that stores an int array's information:
   n: int, the number of elements in the array,
   ir: int*, the pointer to the array,
   seed_time: static time_t, the seed time for the random number generator initialized to time(0).

   Additionally, there are member functions:
   setSeed():static void, sets the seed for the random number generator, using the seed seed_time.
   loadArray():void, dynamically allocates array with new operator.
   printArray():void, prints the content of the array.
   freeArray():void, free the dynamic memory using the delete operator.

   Please place the class in a separate header file with header guards, and its implementation .cpp file.

   In this problem, please make data members all private and member functions all public. The following is the main program you cannot change:

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

int main(){
    RandomArray ra(3);
    RandomArray::setSeed();

    std::cout << "Using array 1: \n";
    for(int i = 0; i < 5; i++){
        ra.loadArray();
        ra.printArray();
        ra.freeArray();
    }

    RandomArray ra2;
    ra2.loadArray();
    std::cout << "Array 2: \n";
    ra2.printArray();
    ra2.freeArray();
    return 0;
}
```

   The following are sample runs:

```
Using array 1:
969     95      732
232     236     402
258     662     971
875     141     262
906     661     701
Array 2:
587     304     604     340     4
```

```
Using array 1:
881     785     136
584     940     813
962     593     622
514     299     726
720     686     846
Array 2:
459     419     268     898     149
```

3. **(20%)** Create a class `OOPClass`, with the following details:

   count: static int, which counts how many objects have been created.
   printCount(): static void, prints the current count value.

   You will need to also have the appropriate constructors. In this problem, please make data members all private and member functions all public. The following is the main program you cannot change:

```
int main(){
    OOPClass a1;
    OOPClass a2;
    OOPClass a3;
    OOPClass::printCount();

    OOPClass a4;
    OOPClass a5;
    OOPClass::printCount();
}
```

   The following is a sample run:

```
Instance of OOPClass created.
Instance of OOPClass created.
Instance of OOPClass created.
Instances of OOPClass: 3
Instance of OOPClass created.
Instance of OOPClass created.
Instances of OOPClass: 5
```

You should try to change the number of objects you create and see of the program can correctly dislay the number of created objects.