Due date: May 4, 11:59 PM

Homework 5

1. Implement the following:

a. Implement a Person class.
b. Dynamic data member - name.
c. Define a function call output.
d. Implement big three.

Use following main() to test your class.

```
int main() {
    cout << endl;
    Person *p = new Person("David");
    p->output();
    Person p1(*p),p2;
    p1.output();
    p2 = p1;
    p2.output();
    delete p;
    cout << endl;
    return 0;
}
```

Output from main function above:

```
Name:David
==> (Person) copy constructor was called.
Name:David
==> (Person) Assignment operator was called
Name:David
==> (Person) Destructor was called

==> (Person) Destructor was called
==> (Person) Destructor was called
```

2. Copy the previous program to a new file, then implement the following:

a. Implement a child class Student that inherits from Person.
b. Dynamic data member in Student class - id
c. override output function.
d. Implement big three.
e. Insure that the output is as specified with the given main function.

Use following main() to test your class.

```
int main() {
    cout << endl;
    Person *p = new Person("David");
    p->output();
    delete p;
    Student s("Jake", 1010);
    p = &s;
    p->output();
    p = new Student(s);
    p->output();
    delete p;
    cout << endl;
    return 0;
}
```

Output from main function above:

```
Name:David
==> (Person) Destructor was called
Name: Jake
ID: 1010
==> (Person) copy constrcutor was called.
==> (Student) copy constrcutor was called.
Name: Jake
ID: 1010
==> (Student) Destructor was called
==> (Person) Destructor was called

==> (Student) Destructor was called
==> (Person) Destructor was called
```

3. Copy the previous program to a new file, then implement the following:

a. Implement a child class Instructor that inherits from Person.
b. Data member in Instructor class - department
c. Override output function.
d. Change Person class to abstract class.

Use following main() to test your class.

```
int main() {
    cout << endl;
    Person **a = new Person*[4];
    a[0] = new Student("David",1212);
    a[1] = new Instructor("Sam","Math");
    a[2] = new Student("Tom",2345);
    a[3] = new Instructor("Jack","History");
```

```
   for(int i=0; i<4; i++) {
      a[i]->output();
      cout<<endl;
   }
   cout << endl;
   return 0;
}
```

Output from main function above:

```
Name: David
ID: 1212

Name: Sam
Department: Math

Name: Tom
ID: 2345

Name: Jack
Department: History
```

4. Implement the following:

   a. A template class named MyArray.

      1) MyArray is a dynamic partially filled array for primitive types.

      2) data members:

      - a pointer for the array
      - any associated variables needed to manage the array.

   3) Constructor must insure that specified capacity is possible. Exit the program if an illegal value is specified.

   4) “The Big Three” are required to insure deep copy.

   5) Private grow function is used to automatically increase the size of the array when adding elements.

   6) Add function to safely append elements to the array.

   7) getSize function that returns the current number of elements.

   8) Overloaded the [] operator to read and update existing elements.

Test using following main function.

```
int main() {
   cout << endl;

   MyArray<int> a(2);
   for(int i=0; i<20; i++) {
      a.add(i+1);
   }
```

```
    a.output();

    MyArray<char> c(20);
    for(int i=0; i<26; i++) {
        c.add(static_cast<char>(i+65));
    }
    c.output();

    cout << endl;
    return 0;
}
```

Output from main function above:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

5. Copy the previous program to a new file and implement the following:

    1) A class SomeObj with a single integer id as a data member.
    2) This class must have a default and user defined constructor.
    3) Create an output function to display the id of the object.
    4) Modify the MyArray class to create an array of SomeObj objects.

Test using following main function.

```
int main() {
    cout << endl;

    MyArray<SomeObj> a(1);

    SomeObj o1(1);
    SomeObj o2(2);
    SomeObj o3(3);
    SomeObj o4(4);
    SomeObj o5(5);
    SomeObj o6(6);
    SomeObj o7(7);
    SomeObj o8(8);
    SomeObj o9(9);
    SomeObj o10(10);

    a.add(o1);
    a.add(o2);
    a.add(o3);
    a.add(o4);
    a.add(o5);
    a.add(o6);
    a.add(o7);
    a.add(o8);
    a.add(o9);
    a.add(o10);

    a.output();

    cout << endl;
    return 0;
}
```

Output from main function above:

```
1 2 3 4 5 6 7 8 9 10
```

Grading policy:

Should submit .cpp file format, other format will be not accepted and assigned 0 point directly.

50% points loss if the program doesn't compile. 50% points for the rest.
If the code compiles and runs, full points if it succeeds for all requirements.