ENSF 480 – Quiz 1 Solutions - OCT 16, 2019

Student Name:_____ **Student ID:**_____

SECTION - I (10 marks) Consider the partial definition of class **Array**, the given main function, and the program's sample output. Then answer questions a, and b.

```
class Array{
    friend ostream& operator << (ostream& os, const Array& a);
public:
  class ArrayIter{
    friend class Array;
   private:
    Array *v;
    int index;
   public:
    ArrayIter (Array& x) {v = &x;}
  };
    Array(int sz);
private:
    int *arrayM;  // points to the first element of array
    int sizeM;     // number of elements in an array
};
//-------------------------------------------------------------------------------
Array::Array(int sz){
  sizeM=sz;
  arrayM = new int [sizeM];
  assert (arrayM != NULL);
}
```

```
void main() {
    Array q = 4;
    Array::ArrayIter itr = q;
    itr[0] = 100;
    itr[1] = 200;
    itr[2] = 300;
    cout << q << endl;
}
```

Program output:

```
100  200  300
```

Question a (5 marks): In the following space convert the given definition of class Array to a template class. **You don't need to write the definition of constructor for class Array.**

The following forward declarations is required because const Array<T> argument in the prototype of operator <<.

```
template <class T>
class Array;
```

// The following forward declarations is required because of friend declaration in class Array

```
template <class T>
ostream& operator << (ostream&, const Array<T>& );

template <class T>
class Array{
public:
  friend ostream& operator<< <T> (ostream& os, const Array<T>& a);
  class ArrayIter{
    friend class Array <T>;
  private:
    Array<T> *v;
    int index;
  public:
    ArrayIter (Array<T>& x) {v = &x;}
  };
  T* arrayM;
  int sizeM;
};
```

Student Name:_____ Student ID:_____

Question b (5 marks): In the following space write the definition of ANY overloaded operators that is needed for the template class, so that the given `main` function works without any error. **You may loose mark for writing unnecessary overloaded operators.**

There are only two operators that MUST be defined to allow the given main function work.

```
template <class T>
T& Array<T>::ArrayIter::operator [](int index) {
    return v->arrayM[index];
}


template <class T>
ostream& operator << (ostream& os, const Array<T>& a) {
    for(int i = 0; i < a.size; i++)
        os << a.arrayM[i] << "  ";
    return os;
}
```

SECTION - II (4 marks)

Consider the partial definition of the following classes. You may assume classes A, B, and C are defined in the same file in the given order from left to right.

```
class A{
public:
    int a1;

protected:
    int a2;
private:
    int a3;
};
```

```
class C: private A{
public:
    int c1;
    void funC();
protected:
    int c2;
private:
    int c3;
};
```

```
class D: public C
{
public:
    int d1;
    void funD();
protected:
    int d2;
private:
    int d3;
};
```

Question: In the following table if a data member is **NOT** accessible in member functions `funC` and `funD`, mark it with (**x**), otherwise leave it blank. Negative marks will be considered for wrong answers.

	a1	a2	a3	c1	c2	c3	d1	d2	d3
funC			**x**				**x**	**x**	**x**
funD	**x**	**x**	**x**			**x**			

SECTION – III (14 marks)

Consider the following C++ listing, then answer the questions a, b, c, d, and e.

```
class A {
protected:
 char *name;
public:
 A(const char* s) {
   name = new char[strlen(s)+1];
   strcpy(name, s);
   cout << endl << "A Ctor" << endl;
 }

 ~A(){ delete [] name;}

 A& operator = (const A& rhs) {
   if(this != &rhs){
       delete [] name;
       name = new char[strlen(rhs.name)+1];
       strcpy(name, rhs.name);
   }
   return *this;
 }

 A (const A& src){
   name = new char [strlen(src.name)+1];
   strcpy(name, src.name);
 }
};
//-------------------------------------------------------------------------------
class B: public virtual A {
 protected:
   char *make;
 public:
   B(const char* s1, const char* s2): A(s1){
     make = new char[strlen(s2)+1];
     strcpy(make, s1);
     cout << endl << "B Ctor" << endl;
   }

   ~B(){  delete [] make; }
};
//-------------------------------------------------------------------------------
class C: public virtual A {
 protected:
   char *make;
 public:
   C(const char* s1, const char* s2): A(s1){
      make = new char[strlen(s2)+1];
      strcpy(make, s1);
      cout << endl << "C Ctor" << endl;
   }

   ~C(){ delete [] make; }
};
//-------------------------------------------------------------------------------
class D: public B, public C{
 protected:
   char *type;
 public:
   D(const char* s1,const char* s2,const char* s3:
                      A(s1), B(s1, s2), C(s1, s2)
   {
      type = new char [strlen(s3)+1];
      strcpy(type, s3);
      cout << endl << "D Ctor" << endl;
   }

   ~D(){ delete [] type;}
};
//-------------------------------------------------------------------------------
int main(void) {
    A *mya = new D("1", "2", "3");
    delete mya;
    D myd("AB", "CD", "ER");
    return 0;
}
```

YOU CAN DETACH THIS PAGE – BUT PLEASE HAND IN WITH THE OTHER PAGES OF THE EXAM.

Student Name:_____ **Student ID:**_____

Question a (2 marks): the above listing causes a runtime error when object **mya** is being deleted in the main function. In the following space explain what is the reason for the error and explain how it should be fixed.

Compiler sees myd as an A object if the destructor is early-bounded. As a result there will be a memory leak because the memory that was dynamically allocated for B part will be leaked.

To solve this problem we should make the class A destructor **"virtual"**

Question b (4 marks): If the issue that is mentioned in question **a**, is properly fixed, **and the first two lines in the main function is commented out**, then what will be the output of this program.

```
A Ctor

B Ctor

C Ctor

D Ctor
```

Question c (2 marks): When the program terminates, what is the order of calls to the destructors of classes.

D, C, B, A

Question d (2 marks): Assuming that the definition of copy constructor for class B and C is given, in the following space write the definition of copy constructor for class D:

```
D::D(const D& src): A(src), B(src), C(src) {
    char *temp = new char[strlen(src.type)+1];
    strcpy(temp, src.type);
    type = temp;
}
```

Question e (4 marks): Assuming that the definition of assignment operators for class B and C are given, in the following space write the definition of assignment operator for class D.

```
D& D::operator= (const D& rhs){
    if(this == &rhs) return *this;
    B::operator = (rhs);
    C::operator = (rhs);
    delete []type;
    type = new char [strlen(rhs.type)+1];
    strcpy(type, rhs.type);
    return *this;
}
```