BLG252E
Object-Oriented Programming
Recitation 1

Tacettin Ayar (ayart@itu.edu.tr)

Doğay Kamar (kamard@itu.edu.tr)

QUESTION 1 (FROM 2015 MIDTERM)

- Write a **swap()** function, which swaps the **two integer parameters** sent from a **main()** function.
- Please also write the **main()** function:
  - You need to **declare two integer values** in the **main()** function. You may initialize them to any value you like.

QUESTION 1 (FROM 2015 MIDTERM)

- Write a **swap()** function, which swaps the **two integer parameters** sent from a **main()** function.
- Please also write the **main()** function:
  - You need to **declare two integer values** in the **main()** function. You may initialize them to any value you like.

```
void swap(int &a, int&b){
   int temp = a;
   a = b;
   b = temp;
}

int main(){
   int x = 3;
   int y = 5;
   swap(x, y);
   cout << x << "\t" << y << endl;
}
```

QUESTION 2 (FROM 2015 MIDTERM)

- The definition of **Class A** and a main program that uses Class A is given below.
- You are expected to analyze the run-time behavior of the given code and **write out the output** generated by the code.

```
class A {
 int a;
 public:
  A();
  A(int);
  A(A&);
  ~A();
  A& operator=(A&);
};
```

```
A::A(){
   cout << "A's Default Constructor has been invoked" << endl;
}

A::A(int a){
   cout<< "A's Constructor has been invoked" << endl;
   this->a = a;
}

A::A(A& inObj){
   cout << "A's Copy Constructor has been invoked" << endl;
   a = inObj.a;
}

A::~A(){ cout << "A's Destructor has been invoked" << endl; }

A& A::operator=(A& inObj){
   cout << "A's assignment op has been invoked" << endl;
   a = inObj.a;
   return *this;
}
```

```
int main(){
   run();
}

void func1(A obj){
   A temp;
   temp = obj;
}

void run(){
   A o1(3);
   A o2 = o1;
   func1(o2);
}
```

ANSWER 2 (FROM 2015 MIDTERM)

- The definition of **Class A** and a main program that uses Class A is given below.
- You are expected to analyze the run-time behavior of the given code and **write out the output** generated by the code.

```
A::A(){
   cout << "A's Default Constructor has been invoked" << endl;
}

A::A(int a){
   cout<< "A's Constructor has been invoked" << endl;
   this->a = a;
}

A::A(A& inObj){
   cout << "A's Copy Constructor has been invoked" << endl;
   a = inObj.a;
}

A::~A(){ cout << "A's Destructor has been invoked" << endl; }

A& A::operator=(A& inObj){
   cout << "A's assignment op has been invoked" << endl;
   a = inObj.a;
   return *this;
}
```

```
int main(){ run(); }

void func1(A obj){
   A temp;
   temp = obj;
}

void run(){
   A o1(3);  A o2 = o1; func1(o2);
}
```

OUTPUT

1. A's Constructor has been invoked
2. A's Copy Constructor has been invoked
3. A's Copy Constructor has been invoked
4. A's Default Constructor has been invoked
5. A's assignment op has been invoked
6. A's Destructor has been invoked
7. A's Destructor has been invoked
8. A's Destructor has been invoked
9. A's Destructor has been invoked

QUESTION 3 (FROM 2017 MIDTERM) (1)

- Carefully inspect the given code. A class named "**FloatingVector**" is created as shown to represent floating point vectors.
- When the following **main** function is called, the given output is produced **without any errors or memory leaks**.
- Inspect the given parts of the class and add missing member methods. **Do not modify** already given members of the class.

```
#include <iostream>
using namespace std;
class FloatingVector{
   int vecSize;
   float * vals;
public:
   FloatingVector (int const = 0,
                   float const * const = NULL);
   void print() const;
   void set(int const, float const);
   // Some other members...
};
FloatingVector::FloatingVector(int const size,
                               float const * const tmp){
   vecSize = tmp==NULL ? 0 : size;
   vals = vecSize==0 ? NULL : new float[vecSize];
   for(int i=0;i<vecSize;i++) vals[i] = tmp[i];
}
```

```
void FloatingVector::print() const {
   cout << "Vector size: "
        << vecSize << " | " << "Content: ";
   for(int i=0;i<vecSize;i++)
      cout << vals[i] << ' ';
   cout << '\n';
}
void FloatingVector::set(int const index,
                         float const val){
   if(index<vecSize) vals[index] = val;
}
```

QUESTION 3 (FROM 2017 MIDTERM) (2)

```
int main(void){
 1: float a_arr[] = {1.1, 2.2, 3.3};
 2: float b_arr[] = {1.2, 2.3, 3.4};
 3: FloatingVector a(3, a_arr);
 4: FloatingVector b(3, b_arr);
 5: FloatingVector c = a + a + b;
 6: a.print();
 7: b.print();
 8: c.print();

 9: float d_arr[] = {0.1, 0.2};
10: FloatingVector d(2, d_arr);
11: FloatingVector e = d;
12: d.set(1, 9.9);
13: d.print();
14: e.print();

 // if the size of the vectors
 // doesn't match return an empty
 // vector of size 0.
15: FloatingVector f = c + d;
16: f.print();
 return 0;
}
```

ANSWER 3 (FROM 2017 MIDTERM) (1)

```
int main(void){
1: float a_arr[] = {1.1, 2.2, 3.3};
2: float b_arr[] = {1.2, 2.3, 3.4};
3: FloatingVector a(3, a_arr);
4: FloatingVector b(3, b_arr);
5: FloatingVector c = a + a + b;
6: a.print();
7: b.print();
8: c.print();

9: float d_arr[] = {0.1, 0.2};
10: FloatingVector d(2, d_arr);
11: FloatingVector e = d;
12: d.set(1, 9.9);
13: d.print();
14: e.print();

 // if the size of the vectors
 // doesn't match return an empty
 // vector of size 0.
15: FloatingVector f = c + d;
16: f.print();
 return 0;
}
```

```
class FloatingVector{
 int vecSize;
 float * vals;
public:
 // ALREADY DEFINED
 FloatingVector (int, float *);     // 3, 4, 10 use
 void print() const;                // 6,7,8,13,14,16 use
 void set(int index, float value); // 12 uses

 // NEWLY DEFINED
 ~FloatingVector(); // give back vals!
 FloatingVector(const FloatingVector &); // Copy Constructor, 11 uses
 FloatingVector operator+(const FloatingVector &); // 5,15 use
};
```

ANSWER 3 (FROM 2017 MIDTERM) (2)

```
FloatingVector::~FloatingVector() { delete[] vals; };

FloatingVector::FloatingVector(const FloatingVector &rhs){
 vecSize = rhs.vecSize;
 vals = new float[vecSize];
 for(int i=0;i<vecSize;i++) vals[i] = rhs.vals[i];
}

FloatingVector FloatingVector::operator+(const FloatingVector &rhs){
 if ( vecSize != rhs.vecSize ) return FloatingVector();
 float *tmpFloat = new float[vecSize];
 for(int i=0;i<vecSize;i++) tmpFloat[i] = vals[i] + rhs.vals[i];
 return FloatingVector(vecSize, tmpFloat);
};
```