ISE103E Sample Questions

Question 1

Analyze run-time behavior of the following code and fill in the table given below with the output generated by each line of the code. If there is no output for a given line, leave the related cell empty.

```
class Aclass{
public:
  Aclass(){
      cout << "Aclass()" << endl;}

  ~Aclass(){
      cout << "~Aclass()" << endl;}

  Aclass(const Aclass &in_c) {
      cout << "Aclass(const &)" << endl;}
};
class Bclass{
  Aclass subpart;
public:
  Bclass(){cout << "Bclass()" << endl;}

  ~Bclass(){
      cout << "~Bclass()" << endl;}

  void func1(Aclass obj){
      cout << "func1" << endl;}

  void func2(){
      Aclass obj;
      cout << "func2" << endl;}
};
```

```
int main(){
Line 1  :    Aclass obj1;
Line 2  :    Aclass *obj2;
Line 3  :    obj2 = new Aclass;
Line 4  :    Bclass obj3;
Line 5  :    obj3.func1(obj1);
Line 6  :    obj3.func2();
Line 7  :    Bclass obj4 = obj3;
Line 8  :    delete obj2;
Line 9  :    return 0;
}
```

Line 1:	**Line 6:**
Line 2:	**Line 7:**
Line 3:	**Line 8:**
Line 4:	**Line 9:**
Line 5:	

Question 2

Design a class to model airplanes (**Airplane**) for an airline company. Each airplane is represented by an integer **id** and the passenger **capacity**, which are given during the creation of an **airplane**. The same **id** may be assigned to different airplanes. If the capacity is not determined during the definition, its value is set to 100.

The automation system of the airline company sets a **Flight** for a specific route (**source** and **destination** airport ids) by assigning an existing **airplane** for the flight. The list of passengers (**psList**), size of which is equal to the capacity of the airplane, is constructed for a specific flight by automatically taking names from the user when the **flight** is defined. To define passenger names, you can use the **String** class presented in the lecture notes. You don't need to write the String class.
The **ontime** status (determines if the flight is on time or delayed) of a flight is defined as a flag (initially a flight is set on time) and can be changed during the flight. A service for printing all attributes of a flight and the airplane information should be provided.

Design the **required** classes with all attributes being private. Provide **only** the required and necessary methods. Make sure that your classes are efficiently and properly designed for public use.

A sample test program is given below:

```
Airplane aip1(1,135);   // Airplane:1, Capacity: 135
aip1.print();
Airplane aip2 = aip1;
Airplane aip3(3);       // Airplane:3, Capacity: 100

Flight fl1(23,45,aip1); // From 23 to 45, by aip1
Flight fl2 = fl1;

fl1.print();        // Prints both flight and airplane
information
fl2.setDelayed();
fl2.print();        // Prints both flight and airplane
information
```