

**ANKARA UNIVERSITY**  
**Computer Engineering Department**  
**COM 102B**  
**Spring 2016-2017**  
**MIDTERM #1**  
**Date:** 05/04/2017  
**Instructor:** Dr. Hacer Yalim Keleş  
**Duration:** 100 mins.

**Student ID:**

**Name & Surname:**

Question #	Total Points	Student Grade
1	10	
2	10	
3	10	
4	20	
5	20	
6	30	
	<b>Grade:</b>	

1- (10 points) Write the output of the following program to the output box.

```
#include <iostream>
#include <string>
using namespace std;

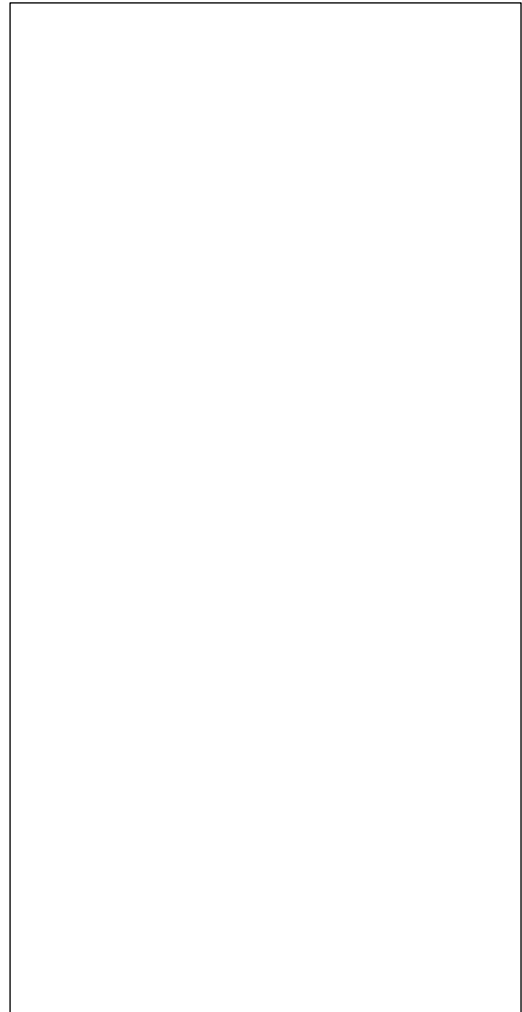
int ff(int& a)
{
    int x = a++;
    return x += a;
}
int gg(int a)
{
    int x = a++;
    return a += x;
}
int hh(int* a)
{
    int x = (*a)++;
    return (*a) + x;
}
int main()
{
    int x = 2;
    int y = 3;
    int z = 4;

    cout<<ff(x)<<"\n";
    cout<<x<<"\n";

    cout<<gg(y)<<"\n";
    cout<<y<<"\n";

    cout<<hh(&z)<<"\n";
    cout<<z<<"\n";
}
```

Output



**2-(10 points)** A set of overloaded function declarations are given below. Write the overloaded version that is called in each case to the boxes next to the function calls. If you think that there is an ambiguity, write -1 inside the corresponding boxes.

```
class Rect
{
    int x;
public:
    Rect(int k=9):x(k){}
};
```

(1) Rect funct(Rect, Rect);

(2) Rect funct(Rect, int);

(3) double funct(double, double);

(4) int funct(int, int);

(5) Rect funct(double, Rect);

(6) Rect funct(Rect, double);

```
void gg(Rect a)
```

```
{
```

```
    funct(2.0, 5.0);
```

```
    funct(3, 3);
```

```
    funct(2, a);
```

```
    funct(a, 5);
```

```
    funct(5.0, 3);
```

```
}
```

**3- (10 points)** Write the ID of the suitable type for each operator definition on the lines next to each function declaration. Notice that there are 4 possible cases as given below. Use the numbers in the paranthesis as the ID of each type, i.e. 1 for prefix unary operator, 3 for binary operator, etc.

**(1) prefix unary (2) postfix unary**

**(3) binary (4) error**

```
class X {
```

```
    X* operator& (); _____
```

```
    X operator& (X); _____
```

```
    X operator++ (int); _____
```

```
    X operator& (X, X); _____
```

```
    X operator/ (); _____
```

```
};
```

```
// nonmember functions :
```

```
X operator- (X); _____
```

```
X operator- (X, X); _____
```

```
X operator- - (X&, int); _____
```

```
X operator- (); _____
```

```
X operator- (X, X, X); _____
```

```
X operator% (X); _____
```

4- (20 points, 10 pts. each) Write the outputs of the following programs to the corresponding boxes shown below.

```
class X
{
    string name;
public:
    X(string s="")
    {
        name = s;
        cout<<"const: "<<name<<"\n";
    }
    ~X()
    {
        cout<<"dest: "<<name<<"\n";
    }
};

X* g()
{
    X a[2];
    X* p = new X("p");
    return p;
}

int main()
{
    X* f;
    X a("a");
    {
        {
            X b("b");
            f = g();
        }
        X c("c");
    }

    delete f;
    X d("d");
}
```

```
class X
{
    static int count;
public:
    X()
    { count++; }
    ~X()
    { count--; }
    static void print()
    {
        cout<<count<<"\n";
    }
};

int X::count = 0;

void f()
{
    X fa[10];
    X* p = &fa[2];
    X::print();
}

void g()
{
    X a;
    {
        X b[2];
        X* c;
        X::print();
    }
    X::print();

    f();
    X c;
    X::print();
}

int main()
{
    X a;
    g();
    X::print();
}
```

5- (20 points) Write the class **Rectangle**, which contains two float values to represent the upper left corner of the rectangle in Cartesian Coordinates and two float values to represent width and height of the rectangle in its data section. Write the following functions using this representation:

- a) (5 pnts) a constructor with default values (0.0) for both coordinates, and (5.0, 5.0) for their width and height.
- b) (10 pnts) overload **operator+** that returns the bounding rectangle of two rectangles .
- c) (5 pnts) overload **operator&** that returns the area of the rectangle.

**e.g.**     Rectangle x;

float b = &x;     // b contains the area, which is 25.0, after this assignment.

```
class Rectangle {
```

6-(30 points) Given the data section and the constructor of the **IntData** below, write the destructor, copy constructor and copy assignment so that it works properly for all kinds of programs *without crashing* and *without causing a memory leak*.

```
class IntData
{
    int* pData;
    int size;
public:
    IntData(int s):size(s)
    {
        pData = new int[size];
    }
```