CS 430 Parameters

1. Suppose a subprogram `void f(int a) { a += 1 }` is called with the statement f(x), where x is a local variable. Draw pictures to show how the value of x is passed between the subprogram and its caller using (a) pass-by-value, (b) pass-by-result, (c) pass-by-value-result, and (d) pass by reference.

pass-by-value: **x -> a (entry, value)**

pass-by-result: **x <- a (exit, value)**

pass-by-value-result: **x -> a (entry, value)**
x <- a (exit, value)

pass-by-reference: **&x -> a (entry, address)**

2. Which parameter passing mechanisms implement *in* mode parameter semantics?

pass-by-value

3. Which parameter passing mechanisms implement *out* mode parameter semantics?

pass-by-result

4. Which parameter passing mechanisms implement *inout* mode parameter semantics?

pass-by-value-result
pass-by-reference
pass-by-name

Consider the following program in a C-like language with static scope.

```
program p
{
  int[] m = [2, 3, 5, 7, 11, 13, 17]
  int x, y = 0, 1;

  void g(int[] a, int b, int c)
  {
    a[b] = 0;
    b += 1;
    y += b;
    a[c] = 8;
  } // end of g

  printline(m[0], m[1], m[2], x, y);
  g(m, x, y);
  printline(m[0], m[1], m[2], x, y);
  g(m, y, y);
  printline(m[0], m[1], m[2], x, y);
}
```

What is the output of this program under the following conditions?

5. All types are value types and all parameters are passed by value.

	p()	first g()	second g()
2 3 5 0 1	m=[2,3,5,…]	a=[~~2~~ 0,~~3~~ 8,5,…]	m=[2,3,~~5~~ ~~0~~ 8,…]
2 3 5 0 2	x=0	b=~~0~~1	b=~~2~~3
2 3 5 0 5	y=~~1~~2~~5~~	c=1	c=2

6. All types are value types and all parameters are passed by value-result. Results are assigned from right-to-left.

	p()	first g()	return	second g()	return
2 3 5 0 1	m=[2,3,5,…]	a = [~~2~~ 0,~~3~~ 8,5,…]	m=[0,8,5,…]	a=[0,~~8~~ ~~0~~ 8,5,…]	m=[0,8,5,…]
0 8 5 1 1	x=0	b=~~0~~ 1	x=1(from b)	b=~~1~~ 2	x=1
0 8 5 1 2	y=~~1~~ 2	c=1	y=~~1(from c)~~ 3	c=1	y=2(from b)

7. All types are value types and all parameters are passed by reference.

	p()	first g()	second g()
2 3 5 0 1	m=[~~2~~ 0,3,~~5~~ ~~8~~ 0,…]	a=&m	a=&m
0 3 8 1 2	x=~~0~~ 1	b=&x	b=&y
0 3 0 1 6	y=~~1~~ ~~2~~ ~~3~~ 6	c=&y	c=&y

8. All types are reference types and all parameters are passed by value.

output same as #7 in this case (no manipulation of references)

9. The int type is a value type, array types are reference types, and all parameters are passed by value.

	p()	first g()	second g()
2 3 5 0 1	m=[~~2~~ 0,~~3~~ 8,~~3~~ ~~0~~ 8,…]	a=&m	a=&m
0 8 5 0 2	x=0	b=~~0~~ 1	b=~~2~~ 3
0 8 8 0 5	y=~~1~~ ~~2~~ 5	c=1	c=2

10. The scalar types are value types, array types are reference types, and all parameters are passed by value-result. Results are assigned from right-to-left.

output same as #6 in this case (no direct manipulation of m from g)