

supervisor	
signature	

* You may answer only in English.

	call-by-value	call-by-reference
advantage	<ul style="list-style-type: none"> safety: call-by-value is often a safe method for argument passing because (a. ...). 	<ul style="list-style-type: none"> The function can (b. ...) the value of the argument variable. Therefore, it is often quite useful. There is no (c. ...) of the argument made. Therefore, it is fast especially when passing large objects.

4. (16 points) We can define a general max function that can take various types. Fill out empty boxes with appropriate C++ codes.

```

(a)
(b) max ( (c) )
{
  return (d) ;
}

```

Output result:
5
5.6
k

5. (18 points) Consider following C++ code. What will be the execution output result? Insert your answer in the empty box below.

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

class A {
public:
    int f() { return 1; }
    virtual int g() { return 2; }
};

class B: public A {
public:
    int f() { return 3; }
    virtual int g() { return 4; }
};

class C: public A {
public:
    virtual int f() { return 7; }
    int g() { return 5; }
};
```

```
int main(){
    A *pa;
    A a;
    B b;
    C c;
    pa=&a; cout << pa->f() << endl; cout << pa->g() << endl;
    pa=&b; cout << pa->f()+pa->g() << endl;
    pa=&c; cout << pa->f() << endl; cout << pa->g() << endl;
    cout << b.f() << endl; cout << b.g() << endl;
    cout << c.f() << endl; cout << c.g() << endl;
    return 0;
}
```

// Execution Output result: insert your answer in this box

6. (24 points) Followings are two different **swap** function implementations that switch two X type input objects. One (left code) uses reference type parameters and the other (right code) uses pointer type parameters. Fill out the empty boxes with appropriate C++ code.

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

class X {
    int z;
public:
    X() { z=0; }
    X(int p) { z=p; }
    int getval() { return z; }
};
```

void swap((a)) // reference type parameters

```
{
    (b)
}
```

```
int main()
{
    X a(1), b(5);

    swap( (c) ); // (*)

    cout << "a=" << a.getval() << endl;
    cout << "b=" << b.getval() << endl;
    return 0;
}
```

ouput result:
a=5
b=1

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

```
class X {
    int z;
public:
    X() { z=0; }
    X(int p) { z=p; }
    int getval() { return z; }
};
```

void swap((d)) // pointer type parameters

```
{
    (e)
}
```

```
int main()
{
    X a(1), b(5);

    swap( (f) ); // (#)

    cout << "a=" << a.getval() << endl;
    cout << "b=" << b.getval() << endl;
    return 0;
}
```

ouput result:
a=5
b=1