

CS 132  
Lambros Petrou @ UCY  
(C / C++)

**PART 1: Find the mistakes:**

**1) ( 1 point )**

```
#include <cstring>
```

```
int main(){
```

```
    char *c;
```

```
    c = "hello";
```

```
    char *s = {'h','e','l','l','o'};
```

```
    for(int i=strlen(c); i>0; i-- )
```

```
        s[i]= c[i];
```

```
    return 0;
```

```
}
```

<- “c” has been allocated 6 bytes and “s” only 5. Therefore this line will be outside of “s” boundaries at the first iteration.

**2) ( 2 points )**

```
void swap(int *a, int *b){
```

```
    int temp = *a;
```

```
    *a=*b;
```

```
    *b=temp;
```

```
}
```

the arguments are pointers to integers so we must DEREFERENCE them in order to change their values.

**3) ( 2 point )**

```
int max(int &a, int &b){
```

```
    return a>b?a:b;
```

```
}
```

& is a reference to an object and not a pointer so we don't need to dereference it. Also if a>b we must return a and not b

**4) ( 3 points ) - Consider including it in more than 1 files**

**// file1.h**

```
#include <iostream>
```

```
using namespace std;
```

before using standard methods state the used Namespace

```
void play();
```

```
int run(){
```

```
    cout << "Run\n";
```

```
}
```

multiple definition of function run()

```
static int x;
```

```
float y=10;
```

multiple definition of variable 'y'

```
extern bool status = true;
```

we cannot assign to a variable declared as extern

**//end of file1.h**

**5) ( 7 points )**

```
class base{
    int i;
public:
    base(int ii):i(ii){}
    ~base(){}
};
```

```
class sub:public base{
    int *tbl;
public:
    sub(int ii):base(ii){
        //i = ii;
        tbl = 0;
    }
    ~sub(){
        if(tbl)
            delete []tbl;
    }
};
```

we must call the constructor of base  
i is private in base class and we don't have direct access

destructor should deallocate any memory at exit otherwise  
there is memory leak

```
void change_tbl(int *t){
    if(tbl)
        delete []tbl;
    tbl = t;
}
};
```

we must check if there is already allocated memory before  
assigning "tbl" to "t" otherwise we have memory leaks

```
int main(){
    sub s;
    for(int i=0; i<10; i++){
        int *t = new int[10];
        s.change_tbl(t);
    }
    s.~sub();
};
```

no sub constructor that gets no arguments

you shouldn't call the destructor yourself, it's  
automatically called at the end of the program

```
base b;
b = s;
s = b;
return 0;
}
```

no base constructor that gets no arguments  
you CAN assign "derived" to "base"  
you CANNOT assign "base" to "derived"

**PART 2: Write what the EXACT output will be in each case assuming you have the following classes and the necessary headers are included correctly.**

```
class base{
public:
    int id;
    base(int i=0):id(i){
        static int c_times = 0;c_times++;
        cout << "base_constructor = " << c_times << endl;
    }

    ~base(){
        static int d_times = 0;d_times++;
        cout << "base_destructor = " << d_times << endl;
    }

    void inc(){
        ++id;
        cout << "new id" << " = " << id << endl;
    }

    virtual void play(){}
    virtual void run(){ cout << "i run" << endl; }
};
```

```
class sub:public base{
public:
    sub():base(){
        static int c_times = 0;c_times++;
        cout << "sub_constructor = " << c_times << endl;
    }

    ~sub(){
        static int d_times = 0;d_times++;
        cout << "sub_destructor = " << d_times << endl;
    }

    void inc(){
        --id;
        cout << "new id" << " = " << id << endl;
    }
};
```

<p><b>1) ( 2 points )</b></p> <pre>base_constructor = 1 base_constructor = 2 base_constructor = 3 sub_constructor = 1 sub_destructor = 1 base_destructor = 1 base_destructor = 2 base_destructor = 3</pre>	<p><b>2) ( 4 points )</b></p> <pre>base_constructor = 1 base_constructor = 2 sub_constructor = 1 new id = 11 new id = 9 sub_destructor = 1 base_destructor = 1 base_destructor = 2</pre>
<p><b>3) ( 4 points )</b></p> <pre>base_constructor = 1 base_constructor = 2 sub_constructor = 1 new id = 1 new id = 1 sub_destructor = 1 base_destructor = 1 base_destructor = 2</pre>	<p><b>4) ( 4 points )</b></p> <pre>base_constructor = 1 base_constructor = 2 sub_constructor = 1 new id = 1 new id = -1 sub_destructor = 1 base_destructor = 1 base_destructor = 2</pre>
<p><b>5) ( 6 points )</b></p> <pre>base_constructor = 1 base_constructor = 2 sub_constructor = 1 i run i run sub_destructor = 1 base_destructor = 1 base_destructor = 2</pre>	<p><b>6) ( 6 points ) - assume 8-bit system is used</b></p> <pre>10 11 f7 80 2 0</pre>

## **PART 3: Programming**

### **1) ( 17 points )**

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

```
class Num{
    int i;
public:
    Num(int ii=0):i(ii){}

    // assignment operator
    Num& operator=(int x){
        i=x;
        return *this;
    }
    Num& operator=(const Num & x){
        if(this != &x)
            i=x.i;
        return *this;
    }

    // equality tests
    bool operator==(const Num & x)const{
        return i==x.i?true:false;
    }
    bool operator!=(const Num & x)const{
        return i!=x.i?true:false;
    }

    // math operations
    Num operator+(const Num & x)const{
        return Num(i+x.i);
    }
    Num operator-(const Num & x)const{
        return Num(i-x.i);
    }
    Num operator*(const Num & x)const{
        return Num(i*x.i);
    }
    Num operator/(const Num & x)const{
        return Num(i/x.i);
    }
    Num& operator+=(const Num & x){
        i+=x.i;
        return *this;
    }
    Num& operator-=(const Num & x){
        i-=x.i;
```

```

    return *this;
}
Num& operator/=(const Num & x){
    i/=x.i;
    return *this;
}
Num& operator*=(const Num & x){
    i*=x.i;
    return *this;
}

```

// increment - decrement

```

Num& operator++(){
    i++;
    return *this;
}
Num operator++(int){
    return Num(i++);
}
Num& operator--(){
    i--;
    return *this;
}
Num operator--(int){
    return Num(i--);
}

```

// returns the actual value of number

```

int val()const{
    return i;
}
};

```

IF you want to test your functions copy the main below

```

int main(){
    Num a(0),b=2,c(a);
    cout << "a= " <<a.val() << " b= " <<b.val()<<" c = " <<c.val()<<endl;
    cout << ((a==c)?"a==c":"a!=c")<<endl;
    cout << ((a!=b)?"a!=b":"a==b")<<endl;
    cout<< "a+b=" << (a+b).val() << " a-b="<<(a-b).val()<< " a*b="<<(a*b).val()<< "
a/b="<<(a/b).val()<<endl;
    cout<< "c+=b " << (c+=b).val() << " c-=b " <<(c-=b).val()<< " a*=b " <<(a*=b).val()<< "
a/=b " <<(a/=b).val()<<endl;
    cout << "++b = " << (++b).val() << " b++ " <<(b++).val()<<endl;
    cout << "--b = " << (--b).val() << " b-- " <<(b--).val()<<endl;

return 0;
}

```