CSCE240 Spring 2017: Exam Review 2 28 Feb 2017

Write the output of the following C++ code segments. Assume everything has been included is correct.

1.

Class.C

```
A::A() {
    cout << "Constructed" <<
endl;
    a = 0;
}

A::A(const A& ref) {
    cout << "Copied" << endl;
    a = ref.a + 1;
}

A::~A() {
    cout << "Deleted" << endl;
    cout << "A is " << a << endl;
}

void A::foo() {
    a += 2;
}

void f1(A a) {
    A b = a;
    b.foo();
    a.foo();
}
void f2(A& a) {
    A *b = &a;
    a.foo();
    (*b).foo();
}
A f3(A a) {
    a.foo();
    A b = a;
    return b;
}
A f4(A a) {
    a.foo();
    A b = a;
    b.foo();
    return b;
}
A& f5(A &a) {
    return a;
}
```

Class.h

```
class A {
    int a;
    public:
        A();
        A(const A&);
        ~A();
        void foo();
};
void f1(A a);
void f2(A& a);
A f3(A a);
A f4(A a);
A& f5(A &a);
```

Main.C

```
int main() {
    A *a = new A();

    f1(*a);
    f2(*a);
    A b;
    b = f3(*a);
    b = f4(b);
    delete a;
    A c = f5(b);
}
```

2.

A.C

```
int a = 3;

A::A() {
    a = 0;
    b = 1;
}
A::A(const A& ref) {
    a = ref.a + ref.b;
    b = ref.b * 2;
}
void A::foo() {
    a++;
    b--;
}
void A::bar() {
    a--;
    b++;
}

ostream& operator<<(ostream& out,
const A &a) {
    out << a.a << " " << a.b;
    return out;
}

void f1(A &a1, A &a2) {
    a--;
    a1.foo();
    a2.bar();
    if (a > 0) {
        f1(a1, a2);
    }
}
```

Main.C

```
int main() {
    A a, b;
    b.foo();
    b = a;
    A c(b);
    f1(c, b);
    f1(a, c);
    cout << a << endl << b <<
endl << c << endl;
}
```

A.h

```
class A {
    int a;
    int b;
    public:
        A();
        A(const A&);
        void foo();
        void bar();
        friend ostream&
operator<<(ostream&, const A&);
};
void f1(A&, A&);
```

3.

Class.C

```
int f1() {
    return 5;
}
int f1(int a) {
    return a*2;
}
int f1(int a, int b, int c) {
    return a + b + c;
}
void f1(char a) {
    cout << a << endl;
}
```

Class.h

```
int f1();
int f1(int);
int f1(int, int, int = 5);
void f1(char);
```

Main.C

```
int main() {
    int a[] = {0,1,2,3,4,5};

    for (int i = 0; i < 6; i++) {
        a[i] = f1(a[i]);
    }
    for (int i = 0; i < 5; i++) {
        a[i] = f1(a[i], a[i +
1]);
    }
    for (int i = 0; i < 6; i++) {
        cout << a[i] << " ";
    }
    cout << endl;
}
```

4.

Main.C

```
int main() {
    int a = 5, b = 6, *c = &a;
    int d, e, f;
    *c = 7;
    d = f1(a);
    e = f2(b);
    f = f3(c);
    cout << a << " " << b << "
" << *c << " " << d << " "
      << e << " " << f << endl;
}
```

Class.C

```
int f1(int & a) {
    return a++;
}
int f2(int a) {
    return a *= 2;
}
int f3(int* a) {
    return *a += 3;
}
```

Class.h

```
int f1(int &);
int f2(int);
int f3(int*);
```

5.

Class.C

```
A::A() {
    cout << "Constructed" <<
endl;
    a = 2;
    b = 5;
}
A::A(const A& ref) {
    cout << "Copied" << endl;
    a = ref.a * 2;
    b = ref.b + 2;
}
A::~A() {
    cout << "Destroyed" << endl;
}
void A::foo() {
    a++;
    b--;
}
void A::bar() {
    a--;
    b++;
}
A f1(A a) {
    a.foo();
    a.bar();
    return a;
}
A f2(A& a) {
    a.foo();
    a.bar();
    return a;
}
A& f3(A& a) {
    a.foo();
    a.bar();
    return a;
}
ostream& operator<<(ostream& out,
const A& a) {
    out << a.a << " " << a.b <<
endl;
    return out;
}
```

Class.h

```
class A {
    int a, b;
    public:
        A();
        A(const A&);
        ~A();
        void foo();
        void bar();
        friend ostream&
operator<<(ostream&, const A&);
};

A f1(A);
A f2(A&);
A& f3(A&);
```

Main.C

```
int main() {
    A a, *b = new A();
    *b = f1(a);
    a = f2(*b);
    *b = f3(a);
    delete b;
    a.foo();
    cout << a << endl;
}
```

6.

Main.C

```
int main() {
    int *a = new int(5), *b = new
int(6), *c = new int(0);
    *c = *a **b;
    cout << *c << endl;
}
```

7.

Class.C

```
Hole::Hole() {
    a = new int[5];
    b = 5;
    for (int i = 0; i < b; i++) {
        a[i] = i;
    }
}
Hole::~Hole() {
    delete [] a;
}
void Hole::foo() {
    delete [] a;
    b++;
    a = new int[b];
    for (int i = 0; i < b; i++) {
        a[i] = i;
    }
}
void Hole::fill(int q) {
    for (int i = 0; i < b; i++) {
        a[i] = q;
    }
}
ostream& operator<<(ostream& out,
const Hole &h) {
    for (int i = 0; i < h.b; i++)
{
        out << h.a[i] << " ";
    }
    out << endl;
    return out;
}
void f(Hole& a, Hole& b) {
    b.b = a.b;
    for (int i = 0; i < a.b; i++)
{
        b.a[i] = a.a[i];
    }
}
```

Class.h

```
class Hole {
    int *a;
    int b;
    public:
        Hole();
        ~Hole();
        void foo();
        void fill(int);
        friend void f(Hole&,
Hole&);
        friend ostream&
operator<<(ostream&, const
Hole&);
};
```

Main.C

```
int main() {
    Hole a, b, c;
    b = a;
    cout << b;
    a.fill(0);
    b.foo();
    cout << a;
    b.fill(0);
    f(b,c);
    c.fill(5);
    b.foo();
    cout << c;
}
```

Find the Errors in the C++ code segments

1.

```
class A {
    int a = 0, b = 1;
    public:
        A();
        friend class B;
}

class B {
    int a, b;
    public:
        B();
}
```

2.

```
class A {
    int a = 0;
    public:
```

```
        A();
        void foo();
}

int main() {
    A a, *b;
    *b = a;
    b.foo();
}
```

3.
```
class A {
    int a = 0;
    public:
        A();
        ~A();
        int foo();
}

int foo() {
    return a;
}
```

4.
```
//*a and *b are of size 10
void copy(const int *a, int *b) {
    for (int i = 0; i < 10; i++)
{
        a[i] = b[i];
    }
}
void f1(const int * a) {
    a++
    *a = 5;
}

void f2(int a) {
    a += 2;
}

void f3(const int &a) {
    a += 5;
}

int f4(int *a) {
    a += 5;
    return a
}
```

5.
```
class A {
    int a;
    int b;
    public:
        A(int z, int x): a(z),
b(x);
        A(const A&);
}

int main() {
    A a(1,5), b(4,9), c;
    c = a + b;
}
```

6.
```
class A {
    int a, b;
    public:
        A();
        ~A();
        ostream& <<(ostream&);
}

ostream& A::<<(ostream& out) {
    cout << a << " " << b;
}
```
7.
```
public void f1(int i) {
    i = 15;
}
public int f1(int i) {
    return (i = 16);
}
private char* f2(char* arr) {
    return *arr;
}
```
8.
```
void f1(int a&) {
    a++;
}

int f1(int a&) {
    a += 2;
    return a;
}
```
9.
```
class A {
    int a = 5;
    int b = 156;
    public:
        A();
        void foo();
}
void copy(const A& ref, const A&
a) {
    a.a = ref.a;
    a.b = ref.b;
}
```
10.
```
void f1(const int * p, int
p_size) {
    p++;
    p[0] = 10;
    *p = 0;
}

void f2(int const * a, int
a_size) {
    a[0] = 55;
    *a = 45;
}

void f3(int * const p, int
p_size) {
    p++;
    *p = 0;
```

```
}

class A {
    public:
        int a, b;
        const int c;
        A(): a(0), b(0), c(105)
{}
        ~A();
        void g()const;
        int g1();
}

void f4(const A &a) {
    a.g();
    a.g1();
}

void A::g()const {
    a++;
    b--;
}

int A::g1() {
    return c + 1;
}
```

11.
```
class A {
    int a[100];
    public:
        A(int q) {for (int i = 0;
i < 100; i++) {a[q] = i;}}
        ~A() {delete [] a;}
        void foo();
}
```

12.
```
class A {
    int *a, a_size;
    public:
        A(int q) {a = new int[q];
a_size = q; for (int i = 0; i <
q; i++){a[i] = q - i;}}
        ~A() {delete a;}
        ostream& operator<<
(ostream&);
}

ostream& A::operator<< (ostream&
out) {
    for (int i = 0; i < a_size;
i++) {
        out << a[i];
    }
    return out;
}
```

Write Code:

1. Write a String class that implements the following features:

   Stores the characters of the string in a character array

   Stores the length of the string

   Functions:

   - String(const char&);
   - ~String();
   - char* getCharArray();
   - int length();
   - char charAt(int index);
   - operator +
   - operator +=
   - operator <<

2. Write a stack that stores ints and implements the following functions:
   - Class Stack:
     - Stack();
     - ~Stack();
     - void push(int);
     - int peek(int);
     - int pop();
     - bool isEmpty();
   - class Node:
     - Node(int, child);

3. Write a class to store a Complex Number (an integer real part and an integer imaginary part) that implements the following functions:
   - Complex(int,int);
   - ~Complex();
   - operator+
   - operator-
   - operator+=
   - operator-=
   - operator<<