

Com S 229 Advanced Programming Techniques

Midterm Exam

11:00am-12:20pm
Tuesday, March 10, 2009

Name: _____

ID (4-digit): ____ _

This is a *closed-book* and *closed-notes* exam. No calculator is allowed.

1	2	3	4	5	Total
48	12	10	14	16	100

1. [48 pts] *Short Questions*

(a) [8 pts] Determine if the following statements are true or false. For each statement, check only the answer you think is correct.

(i) A friend function of a class has access to its private data members.

true _____ false _____

(ii) When the operator `+` is overloaded as a member function, the implicit object is the rightmost operand.

true _____ false _____

(iii) A `bool` value cannot be assigned to an `int` variable because their types are incompatible.

true _____ false _____

(iv) All objects of a class share one copy of each member function.

true _____ false _____

(b) [3 pts] To make one copy of a data member of some class shared by all objects of the class, we declare it using the (marked) keyword

external _____
static _____
const _____
explicit _____

(c) [3 pts] The polynomial $-2x^4 - 4x^2 - 5x + 1$ has

_____ positive roots;
_____ negative roots;
_____ complex roots.

(d) [6 pts] List three situations where a copy constructor is called.

(i)

(ii)

(iii)

(e) [4 pts] What is a memory leak?

(f) [4 pts] What is the value of `*this` inside the implementation of a class member function?

(g) [4 pts] Why does the overloaded assignment operator `=` always return a reference?

- (h) [4 pts] Why does a copy constructor always accept a reference parameter rather than a value parameter? For example, in the `polynomial` class, it has the *first* prototype below but not the second one.

```
polynomial(const polynomial& p);  
polynomial(const polynomial p);
```

- (i) [5 pts] Write a template function `void swap(T a[], T b[], int n);` that swaps the contents of two arrays `a[]` and `b[]` of size `n`. Here `T` can be any predefined data type.

- (j) [7 pts] Write down a few lines of code that allocates dynamic memory for an array `matrix` with sizes 10 and 20 in its two dimensions, respectively.

```
double **matrix;
```

2. [12 pts] *Arrays, Pointers, and References*

Consider five variables declared below:

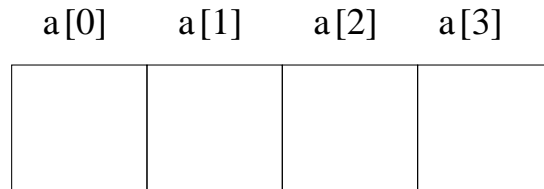
```
int a[ ] = { 4, 5, 3, 2 };
int &b = a[0];
int *c = &(a[2]);
int *&d = c;
int **e = &d;
```

(a) [2 pts] Which of these variables are pointers?

(b) [2 pts] Which of them are references?

(c) [8 pts] Fill the elements in the array **a** (drawn on the next page) after the sequential execution of these six statements:

```
b--;
*d += b + a[1];
c = &(a[1]);
**e -= a[2];
c[2] *= *d + **e + b;
d[1] = 2 * (**e);
```



3. [10 pts] *C++ Class*

Identify the syntax errors in the following class declaration, correct them in place if you could.

```
class big_fat_guy()
{
public
    void big_fat_guy(int a, b, c);
    eat_more(const int* const cheeze; bool beer) const;
    ~big_fat_guy(int d);

private
    int fat[500], size=500;
    double *food;
};
```

4. [14 pts] *Dynamic Memory Management*

The following two questions are based on the code below.

```
template <typename T>
class agent
{
public:
    agent(int v) : size(v)
    {
        value = new T[size];
        cout << size << " allocated." << endl;
    }

    agent(const agent& a) : size(a.size)
    {
        value = new T[size];
        for (int i=0; i<size; ++i) value[i] = a.value[i];
        cout << size << " copy-allocated." << endl;
    }

    agent() : size(1)
    {
        value = new T[1];
        cout << size << " default allocated." << endl;
    }

private:
    T* value;
    int size;
};
```

- (a) [5 pt] Potentially, this class may cause memory leak. Add a member function to the class with outside definition to avoid such a situation.

(b) [9 pt] Write an overloaded assignment operator = outside the class declaration. It should avoid possible dangling references and double deletions.

5. [16 pts] *Dynamic Class*

The code below defines a class `Dynamic` to store a pair of integers.

```
class Dynamic
{
public:
    Dynamic(int m = 1, int n = 1);
    Dynamic(const Dynamic& d);
    ~Dynamic();

private:
    int *pair;
};

Dynamic::Dynamic(int m, int n)
{
    pair = new int [2];
    pair[0] = m;
    pair[1] = n;
}

Dynamic::Dynamic(const Dynamic& d)
{
    pair = new int [2];
    pair[0] = d.pair[0];
    pair[1] = d.pair[1];
}

Dynamic::~~Dynamic()
{
    delete [ ] pair;
}
```

Inside the two functions `main` and `createAnObject` on the next page, comment on all invocations of the constructors and the destructor of the class `Dynamic`. You should put your comments to the right of every statement whose execution either involves or triggers such an invocation. Follow the rules below when writing your comments.

- If an object, say, `foo`, is created with its private data member `pair[] = {1, -1}`, write “object `foo(1, -1)` constructed”.

- If a heap (or free store) object is created with private data member, say `pair[] = {1, -1}`, write “heap object `Dynamic(1, -1)` constructed”.
- Similarly, at the destruction of an object, write “object `foo` destructed” or “heap object `Dynamic(1, -1)` destructed”.
- If no object is constructed or destructed after execution of the statement, leave the comment space blank.

Your comments should indicate clearly which objects are involved.

```

void createAnObject(int m, int n)
{
    Dynamic a(m, n);    //
                       //
}

void main()
{
    Dynamic b = 2;      //
                       //
    Dynamic *p = new Dynamic(3, 3); //
                       //
    Dynamic *q = p;    //
                       //
    Dynamic e = *p;    //
                       //
    createAnObject(2, 3); //
                       //
    p = &e;            //
                       //
    delete q;         //
                       //
}

```