Midterm Examination
(Saturday, January 17, 2004: 9:00am – 10:40am)
(Closed Book, Closed Notes, One A4 sheet allowed)

“I certify that I have neither received nor given un-permitted aid on this examination and that I have reported all such incidents observed by me in which un-permitted aid is given.”

Signature __________________________
Name ______________________________
Student ID __________________________

Question 01 ---------- / [4]
Question 02 ---------- / [8]
Question 03 ---------- / [8]
Question 04 ---------- / [6]
Question 05 ---------- / [4]
Question 06 ---------- / [3]
Question 07 ---------- / [3]
Question 08 ---------- / [6]
Question 09 ---------- / [4]
Question 10 ---------- / [6]
Question 11 ---------- / [5]
Question 12 ---------- / [15]
Question 13 ---------- / [18]
Question 14 ---------- / [10]

Total: ___________/100
Q 1: You are to find the average of some integers. Assume that you store the sum of those integers in an \texttt{int} variable \texttt{sum} and another \texttt{int} variable \texttt{n} contains the total number of integers. You want to store the average in a \texttt{float} \texttt{f}. What statement in old C would accomplish this? What statement would you use if you were to use C++?

Statement in C:

Statement in C++:

Q 2: Consider the following code. Write the output in the box provided on the right.

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

void foo(int x)
{
    int& y = x;
    y++;
}

void bar(int& x)
{
    int& y = x;
    y++;
}

void foobar(int& x)
{
    int y = x;
    y++;
}

int main()
{
    int i = 291;
    cout << i << endl;
    foo(i);
    cout << i << endl;
    bar(i);
    cout << i << endl;
    foobar(i);
    cout << i << endl;
    return 0;
}
```
Q 3: Consider the following two functions:

<table>
<thead>
<tr>
<th>int&amp; foo()</th>
<th>int bar()</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>int i = 3;</td>
<td>int i = 3;</td>
</tr>
<tr>
<td>int&amp; j = i;</td>
<td>int&amp; j = i;</td>
</tr>
<tr>
<td>return j;</td>
<td>return j;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

Answer the following questions:

I) Is it okay to define the function `foo()` as it is done? If not, then why?

II) Is it okay to define the function `bar()` as it is done? If not, then why?

III) Is it okay to do each of the following in a `main()` function? If not, then why?

(a) `int x = foo();`

(b) `int& x = foo();`

(c) `int x = bar();`

(d) `int& y = bar();`
Q 4: In our popular Student class, the name of the student is stored as a C++ string. Not as a pointer and not as a reference; just a plain C++ string. Assume that the class provides a getter as:

    const string& getName() const;

(a) The function could have returned a non-reference; what is the motivation in making the return type a reference? State the reason in one short sentence, no stories!

(b) Assume that we decided that the return type will be a reference for the reason you stated in the above part. Is it absolutely necessary to make the return type a const? Why? Why not?

(c) In part (b) above, whether necessary or not, is there any motivation in making the return type a const?

Q 5: Give at least two restrictions that C++ places on operator overloading?

Q 6: Can we overload + operator as a class method? If not, why?

Q 7: Can we overload insertion operator ("<<") as a class method? If not, why?
Q 8: Increment operator can be overloaded either in the prefix form or in the postfix form. This means that the compiler will call two different functions: one for prefix-increment as in `++obj` and the other for postfix-increment as in `obj++`. How does the compiler know which function to call? Give an example using `BigNum` class. Just provide the declaration of the two functions. We don’t need definitions.

Q 9: Come up with an indigenous definition of generic programming? You just have two sentences. Answers longer than two sentences will not receive any credit.

Q 10: Name three big things which make up the standard template library (STL)?

Q 11: Overloading of insertion operator ("<<") was shown to be taking an `ostream` reference as a parameter. Is it also okay to pass the `ostream` by value? Why? Why not?
Q 12: Consider the following code that tries to keep track of the total number of objects of a class present at a given time. The code compiles and runs but there are logical errors.

```
// file foo.h
#ifndef _FOO_H_
#define _FOO_H_

class Foo
{
    public:
        Foo();
        ~Foo();
        static int getNumInstances()
        { return numInstances; }
    private:
        static int numInstances;
};

#endif /* _FOO_H_ */
```

```
// file foo.cc
#include "foo.h"

int Foo::numInstances = 0;

Foo::Foo()
{
    numInstances++;
}

Foo::~Foo()
{
    numInstances--;
}
```

(a) What is the logical flaw in the above code?

(b) Write a main() function that will expose the flaw in above code:

```
#include "foo.h"

int main()
{
    Foo foo1, foo2, foo3;
    return 0;
}
```

(c) How the flaw can be fixed? Provide code that should go in foo.h and/or foo.cc file.
Q 13: Consider the following class definition which holds an array of integers:

class IntArray
{
    public:
        IntArray(int size);
        IntArray(const IntArray& source);
        const IntArray& operator=(const IntArray& rho);
        int operator[](int index) const;
        int& operator[](int index);
        ~IntArray();
    private:
        int* elements;
        int size;
};

We have declared six class methods in the class definition. Above definition appears in a header file. You are asked to provide the code that must go in the implementation file. That is, write the code for six class methods declared in above class definition. Make sure you include all header files necessary within the implementation file.
Q 14: Consider the following piece of code which depends upon the C++ string class:

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

class A
{
public:
    static const string *vote(string party, string governor, const string *vpp, const string& vp)
    {
        string democrat(*vpp);
        party = democrat;
        governor[1] = governor[0];
        return &vp;
    }

    static void candidates()
    {
        string gore("Al");
        string bush = gore;
        string& nader = bush;
        string buchanan;
        buchanan = gore;
        nader = *vote("Republican", bush, &gore, gore);
    }
};

int main()
{
    A::candidates();
    return 0;
}
```

On the next page, you are required to specify the ordering of all the calls to the string memory management functions. Here are the ones that you will consider:

1. The default constructor
2. The char* constructor
3. The copy constructor
4. The operator= assignment method
5. The destructor

Important:

1. Consider parameter setup when making function calls.
2. Assume that method arguments are always evaluated from left to right. That is, in a call to `foo(CustomString cs, Student s, int i), cs is constructed first, followed by s, followed by i.
3. On the next page, we have provided the start of your solution...
Answer:

Order of calls……

1. `char* constructor for gore`
2.
3.
4.