University of Toronto  
Faculty of Applied Science and Engineering  
ECE 244F  
PROGRAMMING FUNDAMENTALS  
Fall 2013  
Midterm Test  
Examiners: T.S. Abdelrahman, V. Betz, M. Stumm and H. Timorabadi  
Duration: 110 minutes  

This test is OPEN Textbook and CLOSED notes. The use of computing and/or communicating devices is NOT permitted.  

Do not remove any sheets from this test book. Answer all questions in the space provided. No additional sheets are permitted.  

Work independently. The value of each question is indicated. The total value of all questions is 100.  

Write your name and student number in the space below. Do the same on the top of each sheet of this exam book.  

Name:  
(Underline last name)  
Student Number:  

Q1. ________  
Q2. ________  
Q3. ________  
Q4. ________  
Q5. ________  
Q6. ________  
Q7. ________  
Q8. ________  
Q9. ________  
Q10. ________  
Q11. ________  
Q12. ________  

Total
**Question 1. (6 marks). Stringstreams**

A token in a stream is defined as a group of adjacent non-white-space characters surrounded by white space, except for the first token that may not have white space before the first character, and except for the last token that may not have white space after the last character.

You are to write a function, `getNextInt()` that reads from `stringstream ss`. If the next token is an integer, the function should return `TRUE` and the integer; otherwise it should return `FALSE`. Fill in the required code below to define the function `getNextInt()` and fill in the code required to properly make the call to the function.

```cpp
int main() {
    int x;
    stringstream ss;
    ss << getcontent();
    if( getNextInt( ) )
        cout << "The first token is " << x << endl ;
    else
        cout << "Error: first token is not an int" << endl ;
}

bool getNextInt( ) {
```
Question 2. (12 marks). Operator Overloading.

The following class is used to create objects that represent distance, consisting of “feet” and “inches”.

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

class Distance {
  private:
    int feet;
    int inches;

  public:
    Distance(int f, int i);
    int getFeet();
    int getInches();
    void setFeet(int f);
    void setInches(int i);
    void print();
};

Distance::Distance(int f, int i) {
  feet = f;
  inches = i;
}

int Distance::getFeet() {
  return (feet);
}

int Distance::getInches() {
  return (inches);
}

void Distance::setFeet(int f) {
  feet = f;
}

void Distance::setInches(int i) {
  inches = i;
}

void Distance::print() {
  cout << "Distance is: " << feet
       << " feet and" << inches << " inches" << endl;
}
```
We wish to overload the “+” and “==” operators for the Distance class to be able to write code like this in a non-member function (say main):

```cpp
Distance X(10,2);
Distance Y(1,3);
Distance Z(0,0);
:
Z = X + Y;
if (X == Y) ...;
```

For the example declarations above, \(X + Y\) results in an object that represents 11 feet and 5 inches and \((X == Y)\) results in false.

Write the implementation of the two overloaded operator functions as members of the class Distance. Clearly show the function header and its body.

Write the overloaded + operator function below

Write the overloaded == operator function below

The Toronto Census Bureau keeps the records for each of Toronto’s residents in a Resident class. A portion of the declaration for the Resident class is shown below.

```cpp
class Resident {
    public:
        Resident();
    ...
    private:
        string first_name; // e.g. “John”
        string last_name; // e.g. “Baggenstos”
    ...
};
```

The string class allows you to use the < operator to lexically compare two strings. So for example:
- Judy < Suzy
- Jack < Joe
are both true.

For consistency, the symbol < should also be overloaded for the Resident class to sort the residents alphabetically based on their name. As in a standard directory, the names are sorted first by last name and if the last names are the same they are then sorted by first name. For example,
- Judy Baggenstos < Alice Zachoski
- Aaron Baggenstos < Frodo Baggenstos

Write the definition of the operator< member function for the Resident class.

Write the overloaded < operator function below
Question 4. (13 marks). **Constructors and Destructors.**

Consider the following definition and implementation of a class that represents a day of the year. You may assume that this code is **error-free.**

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

class DayOfYear {
private:
    int day;
    int month;
    DayOfYear(const DayOfYear & other);

public:
~DayOfYear();
    DayOfYear(int d, int m);
    int getDay();
    int getMonth();
    void setDay(int d);
    void setMonth(int m);
    DayOfYear operator=(DayOfYear src);
    void print();
};

DayOfYear::~DayOfYear () {/* Nothing to do */}

DayOfYear::DayOfYear (int d, int m) {
    day = d;
    month = m;
}

DayOfYear::DayOfYear(const DayOfYear & other) {
    day = other.day;
    month = other.month;
}

int DayOfYear::getDay() {return (day);}

int DayOfYear::getMonth() {return (month);}

void DayOfYear::setDay(int d) {day = d;}

void DayOfYear::setMonth(int m) {month = m;}

DayOfYear DayOfYear::operator=(DayOfYear src) {
    day = src.day;
    month = src.month;
    return(*this);
}

void DayOfYear::print() {cout << day << “/” << month << endl;}
```
Now consider the following code in the functions `main` and `mystery`. The code is shown in a table to facilitate the writing of your answers. Indicate which lines generate a compile time error by writing `X` next to the line in the second column of the table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Compile-time Error?</th>
</tr>
</thead>
<tbody>
<tr>
<td>using namespace std;</td>
<td></td>
</tr>
<tr>
<td>#include &lt;iostream&gt;</td>
<td></td>
</tr>
<tr>
<td>#include “DayOfYear.h”</td>
<td></td>
</tr>
<tr>
<td>void mystery (DayOfYear);</td>
<td></td>
</tr>
<tr>
<td>int main() {</td>
<td></td>
</tr>
<tr>
<td>DayOfYear FirstOfJuly(1,7);</td>
<td></td>
</tr>
<tr>
<td>DayOfYear Christmas(25,12);</td>
<td></td>
</tr>
<tr>
<td>DayOfYear temp;</td>
<td></td>
</tr>
<tr>
<td>DayOfYear May11(11,5);</td>
<td></td>
</tr>
<tr>
<td>DayOfYear MothersDay = May11;</td>
<td></td>
</tr>
<tr>
<td>DayOfYear today(1,1);</td>
<td></td>
</tr>
<tr>
<td>DayOfYear Oct22(22,10);</td>
<td></td>
</tr>
<tr>
<td>today = Oct22;</td>
<td></td>
</tr>
<tr>
<td>mystery(today);</td>
<td></td>
</tr>
<tr>
<td>return(0);</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>void mystery(DayOfYear source) {</td>
<td></td>
</tr>
<tr>
<td>DayOfYear* temp = new DayOfYear(0,0);</td>
<td></td>
</tr>
<tr>
<td>temp-&gt;setMonth(source.getDay());</td>
<td></td>
</tr>
<tr>
<td>temp-&gt;setDay(source.getMonth());</td>
<td></td>
</tr>
<tr>
<td>delete temp;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>
Question 5. (11 marks). Scopes.

The following class definition describes a simple C++ class called AnExampleClass.

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

class AnExampleClass {
private:
    int value;
public:
    AnExampleClass ()
    {
        value = 0;
        cout << "Constructing object with value 0" << endl;
    }
    AnExampleClass (int v)
    {
        value = v;
        cout << "Constructing object with value " << value << endl;
    }
    ~AnExampleClass ()
    {
        cout << "Destructing object with value " << value << endl;
    }
};

AnExampleClass::AnExampleClass () {
    value = 0;
    cout << "Constructing object with value 0" << endl;
}

AnExampleClass::AnExampleClass (int v) {
    value = v;
    cout << "Constructing object with value " << value << endl;
}

AnExampleClass::~AnExampleClass () {
    cout << "Destructing object with value " << value << endl;
}

Now consider the following code, which uses AnExampleClass:

AnExampleClass a;

void mystery() {
    cout << "Entering mystery" << endl;
    AnExampleClass x(100);
    AnExampleClass* p;
    p = new AnExampleClass(5);
    cout << "Leaving mystery" << endl;
    return;
}

int main() {
    cout << "Starting main" << endl;
    AnExampleClass a;
    if (true) {
        AnExampleClass* p;
        p = new AnExampleClass[2];
        delete [] p;
    } else {
        mystery();
    }
    cout << "Leaving main" << endl;
    return 0;
}
```
In the table below, write the output that an execution of the above program produces in the order in which it is produced. Use one entry in the table for each line of output produced.

<table>
<thead>
<tr>
<th>Output 1</th>
<th>Output 2</th>
<th>Output 3</th>
<th>Output 4</th>
<th>Output 5</th>
<th>Output 6</th>
<th>Output 7</th>
<th>Output 8</th>
<th>Output 9</th>
<th>Output 10</th>
<th>Output 11</th>
<th>Output 12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is the output produced by the following segments of code?

(a)
```cpp
int* p1;
int* p2;
p1 = new int;
p2 = new int;
*p1 = 10;
*p2 = 20;
cout << *p1 << " " << *p2 << endl;
p1 = p2;
cout << *p1 << " " << *p2 << endl;
*p1 = 30;
cout << *p1 << " " << *p2 << endl;
```

Write the output here.

(b)
```cpp
int* p1;
int* p2;
p1 = new int;
p2 = new int;
*p1 = 10;
*p2 = 20;
cout << *p1 << " " << *p2 << endl;
p1 = *p2;
cout << *p1 << " " << *p2 << endl;
*p1 = 30;
cout << *p1 << " " << *p2 << endl;
```

Write the output here.
Question 7. (12 marks). Classes.

Consider the following definition for a class `Mystery`.

```cpp
#include <string>
class Mystery {
    private:
        int value;
    
    public:
        Mystery (const Mystery & s);
        Mystery (const string & s);
        ~Mystery ();
        int getValue() const;
        void setValue(int i);
        bool operator<(const Mystery & rhs);
};
```

Indicate by placing an X in the appropriate column whether each of the following statements is valid, or not valid given the definition of `Mystery` above. A statement is valid if it compiles and executes correctly. A statement is not valid if it either generates a compile-time error, or if it compiles correctly, but produces a run-time error. Assume that each of the statements is independent of the others and that they all appear in the `main` function.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Comments</th>
<th>Valid</th>
<th>Not Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mystery s;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mystery a(sv);</td>
<td>sv is a string object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mystery w(X);</td>
<td>X is an object of type Mystery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mystery c = sv;</td>
<td>sv is a string object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; f.getValue();</td>
<td>f is an object of type Mystery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d &lt; e);</td>
<td>d and e are objects of type Mystery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mystery h = j;</td>
<td>h and j are objects of type Mystery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y = w;</td>
<td>Y and w are objects of type Mystery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e = sv;</td>
<td>e is an object of type Mystery sv is a string object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e &lt; v;</td>
<td>e is an object of type Mystery v is an integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; f.value;</td>
<td>f is an object of type Mystery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f = d + e</td>
<td>f, d and e are objects of type Mystery</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 8. (8 marks). Arrays and Objects.

Consider the following definition of the class Complex:

```cpp
class Complex {
    private:
        float real;
        float imag;

    public:
        Complex();                     // Default constructor
        Complex(float r, float imag);  // Second constructor
        int getReal();
        int getImag();
        void setReal(float r);
        void setImag(float i);
        void print();
    }
```

The following function has been written by a programmer to allocate an array with \( n \) Complex objects, where \( n \) is an integer value read from the input. Complete the code of the function to delete all dynamically allocated data so that there is no memory leak at the end of the function.

```cpp
void create () {
    Complex** p;
    int n;

    cin >> n;

    p = new Complex* [n];

    for (int i=0; i < n; ++i) {
        p[i] = new Complex((float)i,(float)2*i);
    }

    Complex* q = new Complex();
    *q = *p[n-1];
    q->setReal(5.0);
    q->setImag(3.0);

    // ADD CODE HERE
}
```
Question 9. (5 marks). **Debugging and Compilation.**

(a) You are debugging a program in NetBeans, and execution has paused due to hitting a breakpoint at the line highlighted in the figure above. You wish to see what happens when you execute one line of the `getResName` function (i.e. you want to execute just the start of that function, and then examine some variables). Either circle the button you should click on in the screen shot above, or describe the menu command you should use to make this happen.

(b) What is the difference between a compiler error and a compiler warning?

(c) Assuming a file `main.cpp` exists and contains valid C++ code, what is the file that will be generated by the command:

```
g++ -g -Wall main.cpp -o foo
```

(d) Assuming a file `main.cpp` exists and contains valid C++ code, what is the file that will be generated by the command:

```
g++ -g -c main.cpp
```
Question 10. (4 marks). Program Organization.

Consider the program below, which is organized into two header files and three .cpp files.

When compiled with the command:

```
g++ A.cpp B.cpp main.cpp -o prog
```

two compiler errors are reported. Modify some of the files shown above so that the program compiles properly – you can add, delete or modify lines in any of the 5 files. You should make only the minimal modifications necessary in order to make the program compile properly.
Question 11. (7 marks). Functions and Parameters.

(a) (3 marks). Consider the functions below.

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

int compute (int i1) {
    i1 = i1 + 3;
    return (i1);
}

int compute (int i1, int& i2) {
    i1 = i1 + 4;
    i2 = i1 + i2;
    return (i1 + i2);
}

int compute2 (const int& i1) {
    return (i1 + i1);
}

int main () {
    int ival1 = 2;
    int ival2 = 3;
    int resVal = compute (ival1, ival2);
    resVal = resVal + compute (ival2);
    resVal = resVal + compute2 (ival2);
    cout << "resVal is " << resVal << endl;
}
```

Write the program output below.
(b) (4 marks). Consider the following class definition and implementation, contained in a file called “number.cc”. The line numbers shown are for reference and are not part of the code.

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

class Number {
   private:
      int _number;
   public:
      Number ();
      Number (int v);
      void increment () const;
      void copy_and_destroy (const Number & other);
      void print() const;
   };

Number::Number () {
   _number = 0;
}

Number::Number (int num) {
   _number = num;
}

void Number::increment () const {
   _number = _number + 1;
}

void Number::copy_and_destroy (const Number & other) {
   _number = other._number;
   other._number = 0;
}

void Number::print() const {
   cout << _number << endl;
}
```

The above code with the command “g++ -c number.cc” generates some compiler errors. In the table below indicate what these errors are by writing the number of each line of code that generates the error and a brief explanation of the error. There may or may not be as many errors as there are rows in the table.

<table>
<thead>
<tr>
<th>Line # causing error</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Question 12. (10 marks). Programming and File I/O.**

Write a full program, that inputs the name of a text file from cin, opens that file, and then count the number of words and lines in the file. The program should output two numbers: the number of words in the file and the number of lines in the file.

You may assume a file with the input name exists and that the file only contains words with no numbers or isolated punctuation marks. Your program should include the required #include statements. (You can open a file with ifstream filename("filename").)

You will get full marks if the program is written with 25 or fewer statements, where statements counted are if-, while-, for-, and the “#include” statements, as well as any statement ending with “;”. You will get a 10% reduction in your mark for every 2 statements over 25.