University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2011

Midterm Test

Examiner: T.S. Abdelrahman, V. Betz, and M. Stumm

Duration: 110 minutes

This test is OPEN books and OPEN 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 part 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. __________	Q9. __________
Q2. __________	Q10. __________
Q3. __________	Q11. __________
Q4. __________	Q12. __________
Q5. __________	Q13. __________
Q6. __________	Q14. __________
Q7. __________	Q15. __________
Q8. __________	

Total $\boxed{}$

Question 1. (8 marks). *General.*

Answer the following questions by circling either **Yes** or **No**, or by providing a **very brief** and **direct** answer when indicated.

(a) What is the name of the software tool you use in the lab to convert your C++ code into an executable program?

(b) What Unix command would you type to create a new directory called `ece244`?

(c) What is the Unix command needed to make a directory `lab4` accessible only to the owner of the directory?

(d) **Yes** or **No**? An object file (e.g., `main.o`) can be executed by changing its name to `main.exe` and typing the command `./main.exe` at the `Linux` command prompt.

(e) **Yes** or **No**? The `chksubmit` command determines if your program is correct or not by comparing your source files to reference source files. If the files do not match, it issues an error message.

(f) **Yes** or **No**? It is safe for a function to return a pointer to a local variable declared and used inside the function.

(g) **Yes** or **No**? One should always use `delete` to destroy memory allocated with `new` before returning from a function?

(h) **Yes** or **No**? The statement: `delete p`; de-allocates the pointer `p`.

Question 2. (4 marks). *Compilation*.

Suppose you design two classes: `MyFirstClass` and `MySecondClass`. For each of these classes, you have a definition file and an implementation file. Thus, you have four files: `MyFirstClass.h`, `MyFirstClass.cpp`, `MySecondClass.h` and `MySecondClass.cpp`. Also you write a program `main.cpp` that uses the two classes. The files are compiled into a single executable `main.exe`.

Suppose you edit `MyFirstClass.cpp` to change its functionality. Write two different ways in which you can reflect the changes made to `MyFirstClass.cpp` in the executable `main.exe`. Assume you are using the `g++` compiler.

First way:

Second way:

Question 3. (10 marks). *The Make Utility.*

Consider the `Makefile` below.

```
1   targets:            add subtract
2
3   add:                add.o add_fcns.o
4                       g++ add.o add_fcns.o -o add
5
6   subtract:           subtract.o subtract_fcns.o add_fcns.o
7                       g++ subtract.o subtract_fcns.o add_fcns.o -o subtract
8
9   add.o:              add.cc add_fcns.h
10                      g++ -c add.cc
11
12  subtract.o:         subtract.cc subtract_fcns.h
13                      g++ -c subtract.cc
14
15  add_fcns.o:         add_fcns.cc add_fcns.h
16                      g++ -c add_fcns.cc
17
18  subtract_fcns.o:    subtract_fcns.cc subtract_fcns.h
19                      g++ -c subtract_fcns.cc
20
21  clean:
22                      rm *.o add subtract
```

The following table shows several invocations of the `Make` utility using the above correct `Makefile`. For each invocation, indicate the commands that are executed as a result of the invocation, *in the order in which they are invoked*. To simplify providing an answer, the lines of the `Makefile` are numbered as shown above; just indicate the line number corresponding to a command in the table provided below. The invocations of `Make` are in the order shown in the table.

Assume that the Makefile exists in the same directory as the files: `add.cc`, `add_fcns.cc`, `subtract.cc`, `subtract_fcns.cc`, `add_fcns.h`, and `subtract_fcns.h`.

Recall that the `touch` command simply updates the timestamp of its argument to the current time.

Make Invocation	Commands Executed (indicate line number)
make clean	
make	
touch subtract_fcns.h make	
make subtract	
touch add_fcns.h make	

Question 4. (4 marks). *Function Arguments and Pointers.*

Provide two possible ways in which a function `update_ptr` may change the value of a pointer of type `int*`. The pointer is defined outside the function (e.g., `int* ptr;`) and is passed to it as an argument. Assume the formal argument of `update_ptr` is a variable called `p` and that the function is called with an actual argument called `ptr`. Thus, the function changes the value of `ptr`. For each way, show the function prototype and how the function should be invoked.

Write your answer below.

First way:

function prototype: `void update_ptr(                    );`

invocation (i.e., function use): `update_ptr(                    );`

Second way:

function prototype: `void update_ptr(                    );`

invocation (function use): `update_ptr(                    );`

Question 5. (10 marks). *Pointers*

Your task is to examine the code fragments, contained in the table below. If ***any*** of the statements in the code fragment are ***invalid***, place a checkmark in the appropriate column. (For a statement to be "valid", it must be both syntactically and semantically correct.) ***Otherwise***, indicate the value that is printed to the screen. (You can assume that all calls to "cout" succeed.)

Fragment	Invalid?	Value Printed?

`int* y;`
`*y = 6;`
`cout << *y;`		

`int x = 1;`
`int* y = &x;`
`int* z = y;`
`*z = 2;`
`cout << x;`		

`int a[2];`
`a[0] = 1;`
`a[1] = 2;`
`int* c = a;`
`c[1] = 3;`
`a[1] = a[1] + 1;`
`cout << a[1];`		

`int a[2];`
`a[0] = 1;`
`a[1] = 2;`
`int b = 1;`
`int* c = &a[b];`
`b = 0;`
`*c = *c + 1;`
`cout << *c;`		

`int a = 0;`
`int* x = &a;`
`int** y = &x;`
`*x = 1;`
`cout << **y;`		

`int a = 0;`
`int* x = NULL;`
`int** y = &x;`
`x = &a;`
`**y = 3;`
`cout << a;`		

`int a = 1;`
`int b = 2;`
`int* w = &a;`
`int* x = &b;`
`int** y = &w;`
`int** z = &x;`
`*y = x;`
`*w = 3;`
`cout << b;`		

Fragment	Invalid?	Value Printed?

`int a = 1;`
`int* x = &a;`
`{`
`     int a = 2;`
`     int* y = x;`
`     *x = 3;`
`     *y = 4;`
`}`
`cout << a;`		

`int foo (const int &x)`
`{`
`   x = x+1;`
`   return (x+2);`
`}`
 	
`int main ()`
`{`
`   int z = 1;`
`   int y = foo (z);`
`   cout << y;`
`}`		

`int my_func (int &a, int b)`
`{`
`   a = a + b;`
`   b = a + b;`
`   return (a + b);`
`}`
 	
`int main ()`
`{`
`   int i1 = 1, i2 = 2, i3;`
`   i3 = my_func (i1, i2);`
`   i3 += i1 + i2;`
`   cout << i3;`
`}`		

Question 6. (4 marks). *Arrays.*

Identify what is wrong with the following code and then indicate how to fix the error.

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

int MAX = 10;

int list[MAX];

int main() {
    int sum = 0;

   for (int i =0; i <= MAX; ++i)
       list[i] = i;

   for (int i = 0 ; i <= MAX; ++i)
       sum = sum + list[i];

   cout << sum << endl;
}
```

Question 7. (5 marks). *C++ I/O*.

Your task is to extend the command parser you wrote in Lab 3 to add a command, `printmultiple`, which accepts multiple employee numbers and prints out data for each employee. The command format is:

```
printmultiple N emplnum1 emplnum2 ... emplnumN
```

The first parameter, `N`, indicates the number of employee numbers that follow on the line. The following examples are both valid commands:

```
printmultiple 0
Done.
printmultiple 2 547654321 317654320
OUTPUT FOR EMPLOYEE 547654321 APPEARS HERE
OUTPUT FOR EMPLOYEE 317654320 APPEARS HERE
Done.
```

If any of the numbers are invalid (e.g., "`xyz`"), you should **skip processing the remainder of the line without printing an error message**. You may assume there will be no negative employee numbers; only valid positive integers or invalid non-integer input. All inputs are separated by white space (i.e., no "123f"). You may assume that you have access to the following function, which handles the printing of the employee data:

```
void printEmployee (int emplnum);
```

When you are finished processing the `printmultiple` command (whether there were errors or not) you should print "`Done.`". You are **not** expected to handle any other errors, or produce any additional output.

Using only the `cin` operator for input, write a code fragment for recognizing and processing the `printmultiple` command:

Question 8. (4 marks). *Memory Allocation and De-allocation.*

Consider the program fragment below.

```
struct amazing {
    int x;
    int y;
};

// POINT X
{
     struct amazing X;
     int* p;
     struct amazing* another;
     struct amazing* pt = new struct amazing;
     struct amazing* yetanother;

     if (1) {
          struct amazing Y;
          another = pt;
          struct amazing* pt = new struct amazing;
          yetanother = another;
     }

     p = new int [100];

     int i = 0;
     for (i=0 ; i < 100; ++i) p[i]=0;

     // POINT Y
}
// POINT Z
```

Assume no dynamic data exists on the heap at `POINT X`. Indicate below what the programmer must write at `POINT Y` to ensure that no dynamic data exists on the heap at `POINT Z`.

Question 9. (4 marks). *Constructors/Member Methods.*

Consider the following class definition. The numbers listed on the left are not part of the code; they are there for reference.

```
 1: class Golfer {
 2:    private:
 3:       char* fullname;
 4:       int   games;
 5:       int*  scores;
 6:   public:
 7:       Golfer ();
 8:       Golfer (char * name);
 9:       Golfer (char * name, int g);
10:       ~Golfer ();
11: };
```

What class methods (if any) would be invoked by each of the following statements?

Statement	**Class Method** **(write the method's prototype)**
`Golfer nancy;`	
`Golfer lulu ("little lulu");`	
`Golfer roy ("Roy Hobbs", 12);`	
`Golfer *par = new Golfer ();`	

Question 10. (10 marks). *Classes and Objects.*

Consider the following class definition.

```
class BaseC {
    private:
         int a, b;
         void help();
    public:
         int c, d;
         void print();
};
```

The following declarations are made in the `main` function of a program.

```
BaseC base;
BaseC * baseptr = new BaseC();
```

Indicate by placing an **X** in the appropriate column whether each of the following statements is ***correct*** code, or ***incorrect*** code. The statements are also in the `main` function.

	Correct	**Incorrect**
`base.a = 0;`		
`base.c = 0;`		
`base.print();`		
`base.BaseC();`		
`baseptr->nohelp();`		
`BaseC A(0);`		
`this->print();`		
`delete base;`		
`&base = baseptr;`		
`(*baseptr).c = 5;`		

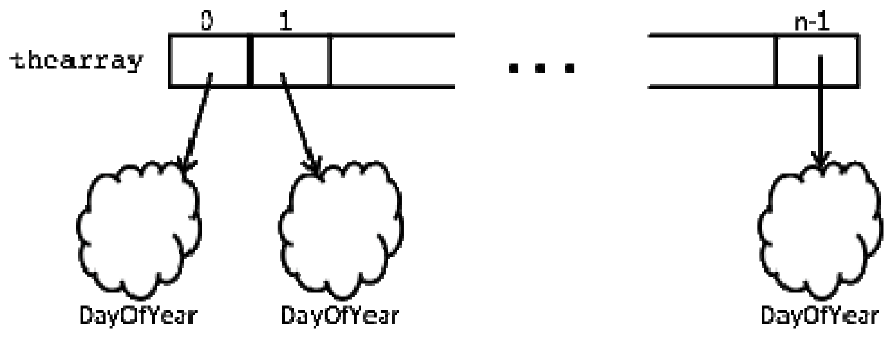
Question 11. (4 marks). *Classes and Objects.*

Write what the program fragment outputs when it is run.

```
class Point {
        int x;
        int y;
    public:
        Point(int i, int j);
        Point increment_x();
        Point increment_y();
        void print();
};

Point::Point(int i, int j) {
     x = i;
     y = j;
}

Point Point::increment_x() {
     ++x;
     return (*this);
}

Point Point::increment_y() {
     ++y;
     return (*this);
}

void Point::print(){
    cout << "(" << x << "," << y  << ")" << endl;
}


int main() {
    Point a(2,3);
    a.increment_x().increment_y().print();
    a.print();
    return (0);
}
```

Question 12. (12 marks). *Dynamic Memory Allocation.*

We wish to allocate the *dynamic* data structure shown below. Given the input `n` from the user, where `n` is a positive integer greater than 1, we wish to dynamically allocate an `n`-element array. Each element points to a dynamically allocated object of type `DayOfYear`.



The definition of the class `DayOfYear` is given below.

```
class DayOfYear {
   private:
       int day;
       int month;
   public:
       void setDay(int d);
       void setMonth(int m);
       void print();
};
```

(a) Give the declaration of the variable `thearray`.

(b) Write code to dynamically allocate the array `thearray`.

(c) Write code to allocate `n` objects of type `DayOfYear` and have each element of `thearray` point to one object.

(d) Write code to de-allocate the object pointed to by the nth element of the array.

(e) Write code to invoke the `setMonth()` method on the 5th element of `thearray`.

(f) Write code to swap the 4th and 5th elements of `thearray`.

Question 13. (8 marks). *Classes/Constructors*

Consider the following partial definition of a C++ class, which is used to store a complex number (*i.e.*, a number with both a *real* and an *imaginary* component):

```
class Complex
{
     private:
          double          real;
          double          imaginary;

     public:
          Complex::Complex()
                    { real = 0.0;  imaginary = 0.0; }

          Complex::Complex(double _r, double _i)
                    { real = _r;  imaginary = _i; }
     ...
     ...
};
```

Complex numbers can also be expressed as a radius ("r") and an angle ("theta"). Assume that you have access to the following functions which convert between "real/imaginary" and "r/theta" representations of a complex number:

```
double     rTheta_to_real (double r, double theta);
double     rTheta_to_imaginary (double r, double theta);

double     realImaginary_to_r (double real, double imaginary);
double     realImaginary_to_theta (double real, double
imaginary);
```

Your task is to extend the *Complex* class to support both forms of imaginary numbers. Write the following new member functions for the *Complex* class:

(a) (2 marks). A constructor which accepts an `r` and a `theta` value, and properly initializes `real` and `imaginary`:

```
     Complex::Complex(double r, double theta)
     {




     }
```

(b) (2 marks). Is it possible to have both the constructor you have just defined, and the `Complex::Complex (double _r, double _i)` constructor in the same class? Why or why not?

(c) **(2 marks)**. Accessor functions which return the `r` and `theta` values which correspond to the stored values `real` and `imaginary`:

```
    double Complex::getR() const
    {







    }
```

```
    double Complex::getTheta() const
    {







    }
```

(c) **(2 marks)**. Recall that, when multiplying two imaginary numbers:

$N_1 = (x_1 + y_1 \boldsymbol{i})$ and $N_2 = (x_2 + y_2 \boldsymbol{i})$

the resulting product, $N_3 = N_1 * N_2$, is calculated as follows:

$N_3 = [\ (x_1{*}x_2 - y_1{*}y_2) + (x_1{*}y_2 + x_2{*}y_1)\ \boldsymbol{i}\ ]$

Your task is to write a member function for *Complex* which implements the multiplication of two *Complex* numbers:

```
    Complex Complex::multiply (Complex & lhs, Complex & rhs)
    {









    }
```

Question 14. (8 marks). *Scopes.*

The following class definition describes a simple C++ class called `sampleClass`.

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

class sampleClass {
private:
        int val;
public:
        sampleClass();
        sampleClass(int v);
        ~sampleClass();
};

sampleClass::sampleClass()
{
        val = 0;
        cout << "Constructing " << val << endl;
}

sampleClass::sampleClass(int v)
{
        val = v;
        cout << "Constructing " << val << endl;
}

sampleClass::~sampleClass()
{
        cout << "Destructing " << val << endl;
}
```

Consider the following code, which uses `sampleClass`:

```
sampleClass a(1);

 void f1()
 {
         sampleClass a[2];
         cout << "Leaving f1()" << endl;
         return;
 }

 sampleClass *f2()
 {
         sampleClass *a = new sampleClass(2);
         cout << "Calling f1()" << endl;
         f1();
         cout << "Leaving f2()" << endl;
         return a;
 }

 int main()
 {
         sampleClass a(3);

         if ((2 + 2) == 4) {
                 cout << "Calling f2" << endl;
                 sampleClass *a = f2();
                 cout << "Back from f2" << endl;
                 delete a;
         }
         cout << "Leaving main" << endl;
         return 0;
 }
```

In the space provided below, write the output that an execution of the above program would produce in the order in which it is produced. Use one entry in the table for each line of output produced.

Question 15. (5 marks). *Programming.*

(a) **(3 marks)**. The code below describes a class to store and manipulate a square matrix of double-precision numbers. Create the definition for the <u>*non-member*</u> function transpose whose declaration is:

```
void transpose (Matrix& my_matrix);
```

Recall that transposing a matrix interchanges its rows and columns. For example, given an initial 3x3 matrix **A**:

$$\mathbf{A} = \begin{vmatrix} 5 & 4 & 3 \\ 8 & 6 & 2 \\ 9 & 10 & 7 \end{vmatrix}$$

Its transpose is:

$$\mathbf{A}^{\mathbf{T}} = \begin{vmatrix} 5 & 8 & 9 \\ 4 & 6 & 10 \\ 3 & 2 & 7 \end{vmatrix}$$

Your transpose function should transpose the matrix **"in place"** – that is, it should modify the passed in matrix object to be the transposed matrix. **No additional storage** (beyond a few local variables) should be allocated during the transpose operation.

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

class Matrix {
   private:
      double **the_matrix;
      int num_rows;  // Square matrix, so num_cols = num_rows
   public:
      Matrix (int num_rows);
      double get_element (int irow, int icol);
      void set_element (int irow, int icol, double value);
      int get_num_rows () {return (num_rows); }
};

void transpose (Matrix &my_matrix);

Matrix::Matrix (int _num_rows) {
   int irow, icol, num_cols, k;

   num_rows = _num_rows;
   num_cols = num_rows;   // Square matrix

   the_matrix = new double *[num_rows];
   for (irow = 0; irow < num_rows; irow++)
      the_matrix[irow] = new double [num_cols];

   k = 0;
   for (irow = 0; irow < num_rows; irow++)
      for (icol = 0; icol < num_cols; icol++)
         the_matrix[irow][icol] = k++;
}
```

```
double Matrix::get_element (int irow, int icol)
{
   return (the_matrix[irow][icol]);
}


void Matrix::set_element (int irow, int icol, double value)
{
   the_matrix[irow][icol] = value;
}
```

(b) (2 marks). Write out the output generated by the main () function listed below.

```
int main () {
   int i, j;
   int N = 5;
   Matrix my_matrix(N);
   transpose(my_matrix);
   for (i = 0; i < N; i++)
   {
      cout << endl;
      for (j = 0; j < N; j++)
         cout << setw(5) << my_matrix.get_element(i, j);
   }
}
```

THIS PAGE IS INTENTIONALLY BLANK FOR ANSWER OVERFLOWS