University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2014

Midterm Test

Examiners: T.S. Abdelrahman and M. Stumm

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. _____	**Q7.** _____
Q2. _____	**Q8.** _____
Q3. _____	**Q9.** _____
Q4. _____	**Q10.** _____
Q5. _____	**Q11.** _____
Q6. _____	**Q12.** _____

Total

Question 1. (10 marks). *Pointers.*

(a) Consider the following declarations

```
int i;
int *pi;
double d;
double *pd;
```

Which of the following statements is valid? Circle all the ones that are valid

```
i=&pi;    *pi=&i;      pd=&pi;      pd=i;      pi=&i;
```

(b) What would be printed from the following C++ code segment?

```
int i=10;
int *p;
int **q;
int ***r;

p=&i;
*p=15;
q=&p;
**q=20;
r=&q;
***r= (*p) + 1;
cout<< i;
```

(c) What is the output of the following C++ code segment?

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

int do_something (int* & ptr1, int* ptr2) {
   int* t;
   *ptr1 = *ptr1 + *ptr2;
   *ptr2 = *ptr1 + *ptr2;
   t = ptr1;
   ptr1 = ptr2;
   ptr2 = ptr1;
   return (*ptr1 + *ptr2);
}

int main () {
   int* p = new int;
   int* q = new int;
   int z;
   *p = 5;
   *q = 8;
   z = do_something (p, q);
   z = z + *p + *q;
   cout << z << endl;
   return (0);
}
```

Question 2. (10 marks). *Functions and Objects.*

(a) Assume there exists four classes: `Mystery`, `StoryLine`, `Novel` and `Author`. We wish to write a member function of class `Author`, called `WriteIt`, which takes two arguments. The first is an object of type `Mystery`, passed by value. The second is a pointer to an object of type `StoryLine`, passed by reference. The function returns an object of type `Novel`, by reference. Write the declaration (prototype) of the function below.

(b) Consider a non-member function called `NoAction`, which takes a single object of type `Time` and returns a single object also of type `Time`. You may assume that the class `Time` is correctly implemented and that `Time.h` is included. Which of the following implementations of this function is correct? Indicate your answer by placing an **X** in the appropriate column in the table.

Implementation	Correct?	Has a problem?

`Time NoAction(Time & t) {`
`  Time temp;`
`  temp = t;`
`  return (t);`
`}`		

`Time NoAction(Time & t) {`
`  Time temp;`
`  temp = t;`
`  return (temp);`
`}`		

`Time & NoAction(Time & t) {`
`  Time temp;`
`  temp = t;`
`  return (t);`
`}`		

`Time & NoAction(Time & t) {`
`  Time temp;`
`  temp = t;`
`  return (temp);`
`}`		

`Time NoAction(Time & t) {`
`  Time* temp = new Time();`
`  *temp = t;`
`  return (temp);`
`}`		

`Time & NoAction(Time & t) {`
`  Time* temp = new Time();`
`  *temp = t;`
`  return (temp);`
`}`		

Question 3. (10 marks). *Classes.*

Consider the following code. It may contain several errors. Identify these lines of code with errors and list them in the table shown on the next page, along with a short explanation of what the error is. To make it easy to refer to lines of code, lines numbers are shown (they are not part of the code). Where a line is not numbered, you may assume it is error-free.

You may not need all the rows of the table. Should there be multiple errors on a line, use multiple entries in the table. You can list the errors in any order.

```
   class PossiblyBad {
     private:
01     int x;
02     int negate();

     public:
03     float y;
04     PossiblyBad(int i, float f);
05     PossiblyBad(PossiblyBad source);
06    ~PossiblyBad(int k);
07     void Mystery(float f);
   };


08 PossiblyBad::PossiblyBad(int i, float f) {
09   x = i;
10 }

11 PossiblyBad::PossiblyBad(PossiblyBad source) {
12   x = source.x; y = source.y;
13 }

14 void PossiblyBad::~PossiblyBad (int k) {
15    x = k;
16 }

17 void PossiblyBad::Mystery(float f) {
18    y = (float) -1*x;
19 }

20 int main () {
21   PossiblyBad a;
22   PossiblyBad b(1);
23   PossiblyBad c(1,3.5);
24   c.x = 2;
25   c.y = 5.8;
26   c.negate();
27   c.Mystery(6.8);
     return (0);
   }
```

Line Number	Explanation of Error

Question 4. (8 marks). *Constructors and Destructors.*

Consider the following definition of a class `Foo`.

```
class Foo {
  private:
     int x;
  public:
     Foo(int i); // call this the Integer constructor
     int getX() const;
     void setX(int i);
     Foo Mystery(Foo one, Foo & two);
};



Foo::Foo(int i) {
    x = i;
}

int Foo::getX() const {
    return x;
}

void Foo::setX(int i) {
    x = i;
}

Foo Foo::Mystery(Foo one, Foo & two) {
   Foo t(8);
   t.x = one.x + two.x;
   two.x = t.x;
   return (*this);
}
```

Consider the code in the main functions below.

```
int main () {
   Foo a(0);
   Foo b(1);

   // point A
   a.Mystery(a,b);
   // point B

   return (0);
}
```

(a) How many times is a constructor of class `Foo` called between points A and B in the code?

(b) Name the constructors that are called between points A and B. Use the table below to name the constructors, one per line.

(c) How many times is a constructor called in the execution of main?

(d) How many times is the destructor called between points A and B in the code?

(e) How many times is the destructor called in the execution of main?

Question 5. (6 marks). *C++ I/O.*

We wish to write a program that prompts the user to enter her first and last names at the standard input and then print her initials to the standard output.

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

int main () {
  char firstInitial;
  char lastInitial;

  cout << "Enter your first name followed by your last name: ";
  ┌──────────────────────────────────────────────────────────┐
  │                                                          │
  │                                                          │
  │                                                          │
  └──────────────────────────────────────────────────────────┘
  cout << "Your initials are: " << firstInitial
       << lastInitial << endl;

  return (0);
}
```

Complete the program by writing code in the box shown above. You are not to declare/use any other variables than the ones shown in the program. However, you may use any of the functions of `iostream` (e.g., `cin.peek()`, `cin.ignore()` or `cin.fail()`). You may also assume that the user will always enter her first name followed by her last name.

Your answer should be **at most 3 lines** of code. You will lose marks for additional lines.

Here are example inputs and outputs for the program (user input is shown in italics):

```
Enter your first name followed by your last name: Patricia Williams
Your initials are PW

Enter your first name followed by your last name: Sandy     Smith
Your initials are SS

Enter your first name followed by your last name: Rachel  McDonalds
Your initials are RM
```

Question 6. (6 marks). *Constructors and Stringstreams.*

Consider the following declaration of a class that represents a day of the year. You may assume that this class is implemented correctly and is **error-free**.

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

class DayOfYear {
   private:
      int day;
      int month;

   public:
      DayOfYear();
      DayOfYear(int d, int m);
      DayOfYear(const DayOfYear & other);
     ~DayOfYear();

      int getDay() const;
      int getMonth() const;

      void setDay(int d);
      void setMonth(int m);

      DayOfYear & operator=(const DayOfYear & src);

      void print() const;
};
```

We wish to be able to write the following code in the function `main`.

```
#include <iostream>
#include <DayOfYear.h"
#include <string>
#include <sstream>

using namespace std;

int main () {
    string s;
    s = "28 10"; //Day is 28 and month is 10, assume no errors in s
    DayOfYear today(s);
    today.print();
    return (0);
}
```

The above code requires demands that one member function be added to the class. Write this function in the space below. Your answer should not exceed a few lines of code.

```
                                                    {




}
```

Write function header here

Write function body here

Question 7. (13 marks). *Arrays and Objects.*

Consider the following definition of the class `Resistor`, which is similar to the one you used in one of your lab assignments. Assume it has been correctly implemented.

```
class Resistor {
   private:
      double resistance;
      string name;
      int endpointNodeIDs[2];
   public:
       Resistor(); // First constructor
       Resistor(double resistance_); // Second constructor
      ~Resistor();
       string getName() const;
       double getResistance() const;
       void setResistance (double resistance_);
       void print ();
};
```

(a) (1 mark). Write a C++ declaration to declare an array called `A` of 20 `Resistor` objects.

(b) (1 mark). Which constructor is used to initialize each of the array elements in the declaration above? Write the name of the constructor (see class definition above), or NONE if no constructor is invoked.

(c) (2 marks). Given an integer variable `n`, write C++ code to dynamically allocate an array called `dA` of `n` `Resistor` objects.

(d) (1 mark). Which constructor is used to initialize each of the array elements in the declaration in part (c) above? Write the name of the constructor (see class definition above), or NONE if no constructor is invoked.

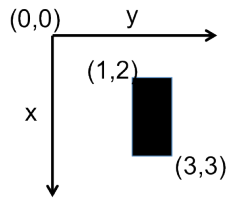
(e) **(2 marks).** Given an integer variable `n`, write C++ code to dynamically allocate an array of `n` pointers to `Resistor` objects. Call the array `dpA`.

(f) **(3 marks)**. Write C++ code to dynamically allocate `n Resistor` objects and have each object pointed to by an element of the array declared in part (e) above. Objects pointed to by even-indexed elements should be initialized using the default constructor. Objects pointed to by odd-indexed elements should be initialized using the second constructor with the resistor value set to the index value. Hint: use the modulus operator %.

(g) **(3 marks)**. Write C++ code to de-allocate the objects and array allocated in parts (e) and **(f)** above so that no memory leaks exist.

Question 8. (8 marks). *Operator Overloading.*

The following class is used to create objects that represent rectangles. Each rectangle is represented by a pair of coordinates: (x_{top},y_{top}) and (x_{bottom},y_{bottom}) corresponding respectively to the x and y coordinates of the top left and bottom right corners of the rectangle (see figure below).




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

class Rectangle {
   private:
      int x_top;
      int y_top;
      int x_bottom;
      int y_bottom;

   public:
      Rectangle(int x_t, int y_t, int x_b, int y_b);

      int getXtop() const;
      int getYtop() const;
      int getXbottom() const;
      int getYbottom() const;

      void setXtop (int x_t);
      void setYtop (int y_t);
      void setXbottom (int x_b);
      void setYbottom (int y_b);

};

Rectangle::Rectangle (int x_t, int y_t, int x_b, int y_b) {
      x_top = x_t;
      y_top = y_t;
      x_bottom = x_b;
      y_bottom = y_b;
}

int Rectangle::getXtop() const {return (x_top);}

int Rectangle::getYtop() const {return (y_top);}

int Rectangle::getXbottom() const {return (x_bottom);}

int Rectangle::getYbottom() const {return (y_bottom);}

void Rectangle::setXtop(int x_t) {x_top = x_t;}

void Rectangle::setYtop(int y_t) {y_top = y_t;}

void Rectangle::setXbottom(int x_b) {x_bottom = x_b;}

void Rectangle::setYbottom(int y_b) {y_bottom = y_b;}
```

We wish to overload the "==" operator for the `Rectangle` class to be able to write code like this in a non-member function (say `main`):

```
Rectangle X(0,0,2,3);
Rectangle Y(4,8,20,10);
   :
if (X == Y) …;
```

The function returns true if the areas of the two rectangles being compared are equal, otherwise returns false. Thus, for the example declarations above, `(X == Y)` returns in false.

Write the implementation of the overloaded `operator==` function, once as a member of `Rectangle` and once as a non-member function. Clearly show the function header and its body.

Write the overloaded `operator==` function as a member of `Rectangle` below

Write the overloaded `operator==` function as a non-member function below

Question 9. (8 marks). *Operator Overloading*.

Joe Programmer wrote a `Str` class to implement strings in a way that uses less memory than the C++ `String` or the C++ `string` class. The `Str` class is to behave just like the `String` class.

For the following lines of code, provide a valid header for the constructor that would be invoked:

```
Str s1 = "abc" ;
```

```
Str  s2 ;
```

```
Str s3 = s2 ;
```

Now, Joe wishes to write concatenate operators so that the following code works as expected:

```
s3 = s1 + s2 ;
s3 = "abc" + s2 ;
s3 = s1 + "bcd" ;
```

How many `operator+` functions does Joe have to define and implement?

How many of those can be a member of the class?

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

Consider the following declaration and implementation of the class `Complex` in the file `Complex.h`.

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

   public:
      Complex();
      Complex(float r, float i);
      float getReal() const;
      float getImag() const;
      void setReal(float r);
      void setImag(float i);
      void print() const;
};
```

Now consider the implementation of the class in `Complex.cpp`.

```
#include "Complex.h"
#include <iostream>
using namespace std;

Complex::Complex() {
    real = new float;
    *real = 0.0;
    imag = new float;
    *imag = 0.0;
}

Complex::Complex(float r, float i) {
    real = new float;
    *real = r;
    imag = new float;
    *imag = i;
}

float Complex::getReal() const {return (*real);}

float Complex::getImag() const {return (*imag);}

void Complex::setReal(float r) {*real = r;}

void Complex::setImag(float i) {*imag = i;}

void Complex::print() const {
      cout << "(" << *real << "," << *imag << ")" << endl;
}
```

Now consider the program below that uses the above `Complex` class.

```
#include <iostream>
#include "Complex.h"
using namepace std;

void flip(Complex x, Complex &y) {
    x.setReal(y.getImag());
    y.setImag(x.getImag());
}

int main() {
    Complex a(5.0,4.6);
    Complex b(3.7,1.5);

    flip(a,b);
    a.print();
    b.print();
    return (0);
}
```

(a) What is the output produced by the program above?

(b) Does the program above suffer from any memory leaks? If so, then how many variables (`int`s, `float`s, or objects of type `Complex`) end up being memory leaks?

(c) There are problems with the above implementation of `Complex` that can be fixed by the addition of three member functions. In the space below, write the implementation of two of these functions (any two). Make sure to include the function header and the function body for each.

Write the first function below

Write the second function below

Question 11. (6 marks). *Constructors*.

Consider the following class definition:

```
#include "B.h"  // Contains the declaration of class B
class A {
     private:
        B b;
        B *bp ;
        int value ;
        :
     public:
        :
}
```

Write a copy constructor that does a deep copy. Include both the constructor's header and its body.

Question 12. (5 marks). *Compilation*.

A program, prog.cpp, makes use of five classes, `A`, `B`, `C`, `D`, and `E` that are defined in the files `A.cpp`, `B.cpp`, `C.cpp`, `D.cpp`, and `E.cpp`, respectively, and declared in their corresponding `.h` files: `A.h`, `B.h`, `C.h`, `D.h`, and `E.h`, respectively. The following table describes which `.h` files are included in which files using the `#include` preprocessor statement:

File	Includes:
`prog.cpp`	`A.h, C.h`
`A.cpp`	`A.h, B.h`
`B.cpp`	`B.h`
`B.h`	`C.h`
`C.cpp`	`C.h`
`D.cpp`	`D.h, E.h`
`E.cpp`	`E.h`

Assume that (*i*) the program has been compiled so that the executable `prog` has been generated along with all of the `.o` files: `A.o`, `B.o`, `C.o`, `D.o`, `E.o`, and `prog.o`, and subsequently (*ii*) the file `C.h` is modified.

List below the precise Unix compiler commands required, in the correct order, to obtain the executable `prog` so that (*i*) the minimal number of `.cpp` files are compiled and so that (*ii*) subsequent changes to the source files allow the compilation of the minimal number of `.cpp` files.