University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2018

Midterm Test

Examiners: T.S. Abdelrahman and M. Mansour

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. __________	**Q8.** _________
Q2. __________	**Q9.** _________
Q3. __________	**Q10.** _________
Q4. __________	**Q11.** _________
Q5. __________	**Q12.** _________
Q6. __________	**Q13.** _________
Q7. __________	**Q14.** _________

Total

Question 1. (9 marks). *General.*

Answer the following questions by circling either **TRUE** or **FALSE**.

(a) **TRUE** or **FALSE**. Two different class definitions may have the same member names.

(b) **TRUE** or **FALSE**. An object can only be passed to a function as a call-by-reference parameter.

(c) **TRUE** or **FALSE**. A class member function may be private.

(d) **TRUE** or **FALSE**. All constructors for a class must be public.

(e) **TRUE** or **FALSE**. It is possible to have multiple private labels in a class definition.

(f) **TRUE** or **FALSE**. When memory is allocated using the following statement:

```
int** p = new int*[10];
```

then it can be de-allocated or freed using:

```
delete [] *p;
```

(g) **TRUE** or **FALSE**. You must write a copy constructor function for every class you create if you wish to initialize an object from another of the same type.

Answer the following questions by circling the most appropriate answer.

(h) In a class, all members are ____________ by default

a. public
b. private
c. global
d. None of the above

(i) Operators can be overloaded as

a. Friends of a class
b. Members of a class
c. Non-friends, non-members of a class
d. All of the above

(j) If we have accessor and mutator members of a class, why would we have friend functions?

a. We should not have them
b. For more efficient access to the private data members.
c. Friend functions must call the accessor or mutator functions anyway.
d. None of the above

Question 2. (8 marks). *Separate Compilation.*

You are given a program that has a main function and 3 classes: `First`, `Second` and `Third`. For each of these classes, you have a definition file and an implementation file. Thus, you have seven files in total: `First.h`, `First.cpp`, `Second.h`, `Second.cpp`, `Third.h`, `Third.cpp` and `main.cpp`. All the files exist in the same directory. The first few lines of each file are shown below. The rest of the contents of each file is irrelevant to the question and is shown as "`:`". You may assume the definition/implementation files are error-free.

First.h

```
#ifndef FIRST_H
#define FIRST_H

class First {
  :
};

#endif
```

Second.h

```
#ifndef SECOND_H
#define SECOND_H

class Second {
    :
};

#endif
```

Third.h

```
#ifndef THIRD_H
#define THIRD_H

class Third {
    :
};

#endif
```

First.cpp

```
#include “First.h”

First::First () {
    :
}
    :
```

Second.cpp

```
#include “First.h”
#include “Second.h”

Second::Second ()
{
    :
}
    :
```

Third.cpp

```
#include “Second.h”

Third::Third ()
{
    :
}
    :
```

main.cpp

```
#include “First.h”
#include “Second.h”
#include “Third.h”


int main () {
    :
    :
}
```

The files are to be *separately* compiled and then linked into a single executable called `main.exe`.

(a) **(3 marks)**. Write down the Unix commands necessary to *separately* compile the above files and generate the executable.

Write your answer here

(b) **(2 marks)**. You modify the file `Second.cpp`. Write down the Unix commands necessary to regenerate the executable by compiling the smallest number of files possible.

Write your answer here

(c) **(3 marks)**. You modify the file `First.h`. Write down the Unix commands necessary to regenerate the executable by compiling the smallest number of files possible.

Write your answer here

Question 3. (9 marks). *C++ I/O.*

Consider the following program that uses `stringstreams` to read a command. The command has the following format:

```
count intArg
```

The command word is `count` and `intArg` is an integer argument. The command must have only <u>one</u> integer argument.

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

#include <sstream>
#include <string>

// function prototype

bool handle_count(                                        );

int main() {
   string line;
   string command;
   int intArg;

   getline(cin, line);
   stringstream lineStream(line);

   lineStream >> command;
   if (command == "count") {

      if (handle_count(                          )  ) {

          cout << "Integer argument is " << intArg << endl;
          return (0);
      }
      else {
         cout << "Invalid arguments" << endl;
         return (-1);
      }
   }
   else {
      cout << "Invalid command" << endl;
      return (-1);
   }
}
```

Write your answers here

The function `handle_count` performs the reading of the integer value. If the integer is valid, it returns `true` and updates the value of `intArg`. Otherwise, it returns `false`.

(a) (4 marks). Determine the number of arguments and the type of each argument and indicate them in the code above in the prototype of the function. Further, indicate what parameters are passed to the function when it is invoked. Write your answers where indicated in the code above.

You **may not modify** `main` by adding or removing line, other than by indicating the formal arguments in the function prototype and actual arguments of the function invocation.

(b) (5 marks). Write the header and body of the `handle_count` function below so it performs as indicated above. Write your answer in the box below.

```
bool handle_count(                                    ) {














}
```

Question 4. (4 marks). *Pointers.*

Consider the following `main` function. The line numbers to the left are for reference and are not part of the code.

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

01  int main () {
02    int*  first_ptr;
03    int*  second_ptr;
04    int** p_ptr;
05    first_ptr = new int;
06    second_ptr = new int;
07    p_ptr = &first_ptr;
08    *first_ptr = 4 ;
09    *second_ptr = 8 ;
10    second_ptr = *p_ptr;
11    cout << *first_ptr << " " << *second_ptr << endl ;
12    delete first_ptr;
13    delete second_ptr;
14    delete *p_ptr;
15    return (0);
16  }
```

(a) (2 marks). What is the output produced by by `cout` on line 11 of the code.

Answer:

(b) (2 marks). The program **may** have a problem with it. What is the problem, if any? Circle only one answer.

A. The program has no problem with it.
B. The program has a memory leak.
C. The delete on line 14 should not dereference `p_ptr`, but use it directly.
D. The program deletes the same region of memory more than once.
E. B and C.
F. B and D.
G. B, C and D.

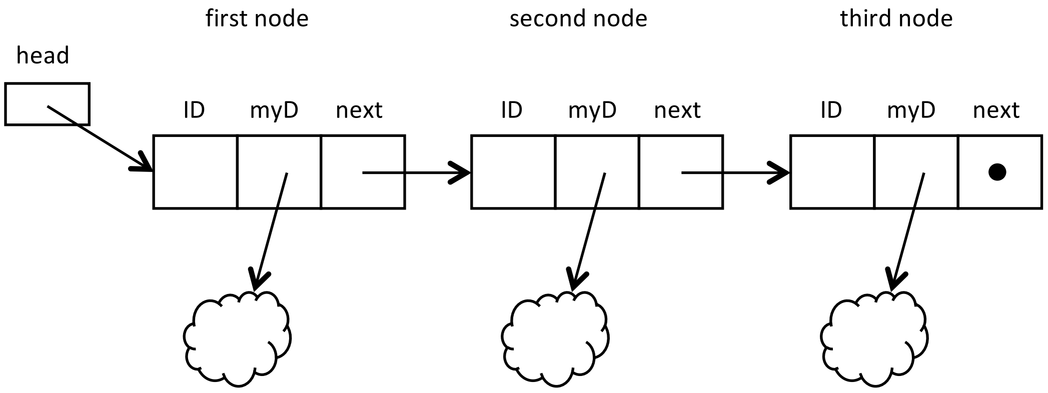
Question 5. (7 marks). *Pointers and Classes.*

Consider the following class declaration.

```
class node {
  public:
     int   ID;
     Data* myD;
     node* next;
};

node* head;
```

The definition of the class `Data` is not shown here, but you may assume it to exist and to be correct. Dynamic data is allocated in the `main` function of a program. This dynamic data is depicted in the figure below.



(a) (3 marks). Using only one integer variable called `temp`, write lines of code to swap the `ID` members of the first and third nodes shown in the figure above. No other variables are allowed.

Write your answer here.

```
int temp;
```

(b) (4 marks). Using only one pointer variable called `temp_ptr`, write lines of code to swap the second and third nodes shown in the figure. You are not to use any other variables and you must swap the nodes not the members of the nodes.

Write your answer here.

```
node* temp_ptr;
```

Question 6. (5 marks). *Dynamic Memory Allocation.*

Study the following program and answer the questions below. You may assume that the class `Mystery` is correctly defined an implemented.

```
#include "Mystery.h"
void level_B (Mystery obj) {
   Mystery* ptr = new Mystery();
   Mystery x[2];
   // Point Z
   return;
}

void level_A (Mystery* x) {
    Mystery a;
    Mystery b;
    // Point Y
    Mystery* ptr = new Mystery[10];
    delete [] x;
    // Point W
    return;
}

int main() {
    Mystery a;
    Mystery* b;
    // Point X
    b = new Mystery[2];
    level_A(b);
    level_B(a);
    // Point Q
    return (0);
}
```

(a) Indicate the number of objects of type `Mystery` that exist in memory when the program execution reaches `Point X`.

Answer:

(b) Indicate the **change** in the number of objects of type `Mystery` that exist in memory and that occurs during program execution between `Point X` and `Point Y`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer:

(c) Indicate the **change** in the number of objects of type `Mystery` that exist in memory and that occurs during program execution between `Point Y` and `Point W`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer:

(d) Indicate the **change** in the number of objects of type `Mystery` that exist in memory and that occurs during program execution between `Point W` and `Point Z`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer:

(e) Indicate the **change** in the number of objects of type `Mystery` that exist in memory and that occurs during program execution between `Point Z` and `Point Q`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer:

Hint: Draw a picture!

Question 7. (6 marks). *Classes.*

The following is the definition/implementation of a class called `Foo`.

```
class Foo {
  private:
    int priv;

  public:
    Foo (int pv) {priv = pv;}

    Foo (const Foo src) {priv = src.priv;}

    Foo& operator=(Foo& rhs) const {
      priv = rhs.priv;
      return this;
    }

    int  getPriv() const { return priv; }
    void setPriv(int pv) { priv = pv; }
};
```

Compiling the above definition/implementation results in one or more errors. Re-write the class so it is error-free. Write your answer (the entire definition/implementation) in the box below. You may want to underline in your answer below the changes you made.

Question 8. (10 marks). *Classes and Objects.*

Consider the following class definition and implementation.

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

class Shape {
    int x_pos;
    int y_pos;
  public:
           Shape (int x, int y);          // Method 1
           Shape (const Shape& source);   // Method 2
          ~Shape ();                      // Method 3
    Shape& operator= (Shape rhs);         // Method 4
    void   print() const;                 // Method 5
};

Shape::Shape(int x, int y) {
    x_pos = x;
    y_pos = y;
}

Shape::Shape(const Shape& source) {
    x_pos = source.x_pos;
    y_pos = source.y_pos;
}

Shape & Shape::operator=(Shape rhs) {
    x_pos = rhs.x_pos;
    y_pos = rhs.y_pos;
    return (*this);
}

void Shape::print() const {
    cout << "(" << x_pos << "," << y_pos  << ")" << endl;
}
```

For each of the following snippets of code, assumed to be in the `main` function of the program, indicate which methods are invoked. Circle **all** correct answers.

Note: while correct answers get full marks, some incorrect answers get part marks (and some get none).

1. `Shape circle (5,10);`

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

2. `Shape rectangle;`

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

3. `Shape* triangle;`

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

4. `Shape* hexagon = new Shape[100];`

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

5.
```
Shape circle1 (3,4);
Shape circle2 (circle1);
```

   a. None
   b. Method 1
   c. Methods 1 and 2
   d. Methods 1 and 3
   e. Methods 1 and 4
   f. Methods 2, 3 and 4
   g. The code snippet results in a compile time error

6.
```
Shape circle3 (3,4);
Shape circle4 = circle1;
```

   a. None
   b. Method 1
   c. Methods 1 and 2
   d. Methods 1 and 3
   e. Methods 1 and 4
   f. Methods 2, 3 and 4
   g. The code snippet results in a compile time error

7.
```
Shape circle5 (3,4);
Shape circle6 (6,8);
circle5 = circle6;
```

   a. None
   b. Method 1
   c. Methods 1 and 2
   d. Methods 1 and 3
   e. Methods 1, 2 and 4
   f. Methods 1, 2, 3 and 4
   g. The code snippet results in a compile time error

Question 9. (8 marks). *Constructors and Destructor.*

Consider the following definition/implementation of a class called `Shape` that appears in the file: `Shape.h`.

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

class Shape {
  private:
    int ID;

  public:
    Shape () {
      ID = 0;
      cout << "Constructor 1 " << ID << endl;
    }

    Shape (int id) {
      ID = id;
      cout << "Constructor 2 " << ID << endl;
    }

    Shape (const Shape& s) {
      ID = s.ID;
      cout << "Constructor 3 " << ID << endl;
    }

    ~Shape() { cout << "Destructor " << endl;}

    Shape& operator=(Shape rhs) {
      cout << "Operator= " << ID << endl;
      ID = rhs.ID;
      return *this;
    }

    int  getID() const { return ID; }
    void setID(int id) { ID = id; }
};
```

The following is a main program that uses the above class.

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

#include “Shape.h”

int getID (Shape s) {return s.getID();}
int getShapeID(Shape& s) {return s.getID();}

int main () {
  Shape circle(5);
  Shape* line1[2];
  Shape* polygon = new Shape(circle);
  line1[0] = polygon;
  line1[1] = polygon;
  *polygon = circle;
  Shape line2[2];
  line2[0] = *polygon;
  cout << *(line1[0]).getID() << endl;
  cout << getID(*line1[1]) << endl;
  cout << getShapeID(*line1[0]) << endl;
  return 0;
}
```

What is the output of the program? Select one of the answers from the table below.

Write one of `A`, `B`, `C`, `D`, `E`, `F` or `G` here:

Note: There is only one correct answer that receives the full mark. However, incorrect answers do get part marks (some get more than others and some get none).

A	Constructor 2 5 Constructor 3 5 Constructor 3 5 Operator= 5 Destructor Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Destructor 5 5 5 Destructor Destructor Destructor	D	Constructor 2 5 Destructor Constructor 1 0 Constructor 1 0 Destructor 5 5 Destructor 5 Destructor Destructor Destructor
B	Constructor 2 5 Constructor 3 5 Constructor 3 5 Operator= 5 Destructor Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Destructor 5 Constructor 3 5 5 Destructor 5 Destructor Destructor Destructor	E	Constructor 2 5 Constructor 3 5 Constructor 3 5 Operator= 5 Destructor Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Destructor 5 Constructor 3 5 5 Destructor Constructor 3 5 5 Destructor Destructor Destructor Destructor

C	Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Constructor 3 5 Operator= 0 Destructor Constructor 3 5 Operator= 0 Destructor Constructor 3 5 Operator= 5 Destructor Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Destructor 5 Constructor 3 5 5 Destructor 5 Destructor Destructor Destructor Destructor Destructor	F	Constructor 2 5 Constructor 3 5 Constructor 3 5 Destructor Constructor 1 0 Constructor 1 0 Constructor 3 5 Destructor 5 Constructor 3 5 5 Destructor 5 Destructor Destructor Destructor
G	None of the above		

Question 10. (8 marks). *Class Definition*.

Consider the following definition and implementation of a class `aVector` that represents vectors of integers. Consider also the short `main` function that makes use of the class. You may assume they are all correctly defined and implemented.

```
#define max_size 64
class aVector {
   private:
    int size;                // Size of vector
    int theVector[max_size]; // Vector elements from 0 to size-1
   public:
    int getSize() const {return size;}
    void setSize(int sz) {size = sz;} // Should be used only once
    int  getElement(int i) {return theVector[i];}
    void setElement(int i, int v) {theVector[i]=v;}
};

int main() {
   aVector v1;
   v1.setSize(10); // Used once to set size of vector
   aVector v2(v1);
   aVector v3;
   v3 = v1;
   return (0);
}
```

The creator of the class decides to change its definition to allow for exact-sized vectors. Thus, the definition of the class is changed to:

```
class aVector {
   private:
    int size;                // Size of vector
    // pointer to dynamically allocated elements from 0 to size-1
    int* theVector;
   public:
    int getSize() const {return size;}
    void setSize(int sz) {size = sz;} // Should be used only once
    int  getElement(int i) {return theVector[i];}
    void setElement(int i, int v) {theVector[i]=v;}
};
```

Note the absence of `max_size`. The intent of the programmer is to have `theVector` pointer private member to point to a dynamically allocated the array of exactly `size` integers, containing vector elements of the vector.

Answer the following questions, keeping in mind that <u>marks will be subtracted for incorrect answers</u>.

(a) **(2 marks)**. What additional (i.e., new) class member functions must be defined for the class to remain correct? Write only the prototypes of these functions. If no new members must be added, write **NONE**.

(b) **(2 marks)**. What existing member functions must be removed for the class to remain correct? Write only the prototypes of these functions. If no members must be removed, write **NONE**.

(c) **(2 marks)**. Which of the functions that were not removed (in part (b)) from the original class definition must be re-implemented to reflect the new definition of the class? Again, write only the prototypes of these functions. If none must be re-implemented, write **NONE**.

(2 marks). Identify the lines in main must be changed for main to continue to work correctly with the changes made to the class Complex. For each line that must change, write down the original line and what it should be changed into. If no lines must change, write down **NONE**.

Question 11. (2 marks). *Classes and Objects.*

Once upon a time, in a land far away, there was a very good C++ programmer. Her name was Amy. She spent a few days designing and implementing a C++ class called `Mystery`. The following is the class definition that appears in `Mystery.h`:

```
#include "YetAnotherMystery.h"
class Mystery {
   private:
      YetAnotherMystery* mystery_variable;

   public:
      Mystery();
      Mystery(const Mystery & other);
      ~Mystery();
      Mystery & operator=(const Mystery & rhs);
      void operateOnMystery();
      void print () const;
};
```

Amy is so proud of her class that she does not want objects of type `Mystery` to ever be deleted. In other words, so wants code that looks like this:

```
{
     Mystery a;
}
```

in a `main` function to not compile since the object a will be deleted when it goes out of scope. The same for code that looks like this since "`delete p`" will delete the dynamically-allocated object:

```
Mystery * p;
p = new Mystery ();
:
delete p;
```

In <u>**one short**</u> sentence, describe what Amy can do to her class definition shown above to accomplish her goal of preventing objects of type `Mystery` from getting deleted.

Question 12. (8 marks). *Arrays and Objects.*

Consider the following definition/implementation of the class `DayOfYear`. You may assume the class is correctly defined/implemented.

```
class DayOfYear {
   private:
       int day;
       int month;
   public:
           DayOfYear(int d, int m) {day = d; month = m;}
          ~DayOfYear() {}
       int getDay() const {return day;}
       int getMonth() const {return month;}
       void setDay(int d) {day = d;}
       void setMonth(int m) {month = m;}
};
```

The following `main` function dynamically allocates then de-allocates `n` `DayOfYear` objects, along with other variables, where `n` is an integer value read from `cin`. 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. Assume `iostream` has been included and that the `std` namespace is used.

```
int main () {
   DayOfYear*** p;
   int n;
   cin >> n;
   p = new DayOfYear** [n];

   for (int i=0; i < n; ++i) {
       p[i] = new DayOfYear*;
       *(p[i]) = new DayOfYear(i%30, i%12);
   }

   DayOfYear* q = *p[0];
   q->setDay(1);
   q->setMonth(1);

   // ADD YOUR CODE HERE






   return (0);

}
```

Question 13. (8 marks). *Operator Overloading*.

Consider the following definition and implementation of a class `myVector` that represents vectors of integers. You may assume they are correctly defined and implemented.

```
#define max_size 256

class myVector {
   private:
    int size;                // Actual size of myVector
    int theVector[max_size]; // myVector elements from 0 to size-1
   public:
    myVector(int sz);
    ~myVector();
    int getSize() const {return size;}
    void setSize(int sz) {size = sz;}  // No error checking, but OK
    int  getElement(int i) {return theVector[i];}
    void setElement(int i, int v) {theVector[i]=v;}
};
```

(a) (4 marks). Overload the plus operator (+) as a member of the class my`Vector` to perform the addition of two vectors. The addition of a vector `A` to a vector `B` produces <u>a new vector</u> whose elements are the element-wise addition of `A` and `B`. For example, if `A` = `(1 4)` and `B` = `(2 3)`, then the result of `A + B` is the vector `(3 7)`. For simplicity, you may assume that the two vectors added will always have the same size.

Write your answer in the box below. Be sure to indicate both the header and the body of the method.

{ ← Write function header here

← Write function body here

}

(b) (4 marks). Re-write the above overloaded operator so it is not a member function of the class `Vector`. You may not change the definition of the class in any way.

Write your answer in the box below. Be sure to indicate both the header and the body of the method.

```
                                                              {
}
```

Write function header here

Write function body here

Question 14. (8 marks). *Objects with Pointers*.

Consider the following definition and implementation of a class `Mystery`. The definition and implementation of the classes `Left` and `Right` are not available but you may assume they are correctly defined and implemented.

```
#include "Left.h"
#include "Right.h"

class Mystery {
   private:
     Left*  leftside;
     Right* rightside;

   public:
     Mystery ();
     ~Mystery ();
     Left* getLeft() const;
     Right* getRight() const;
};

Mystery::Mystery() {
   leftside = new Left();
   rightside = new Right();
}

Mystery::~Mystery() {
   delete leftside;
   delete rightside;
}

Left* Mystery::getLeft() const {return leftside;}

Right* Mystery::getRight() const {return rightside;}
```

Now consider the following segment of code in the `main` function.

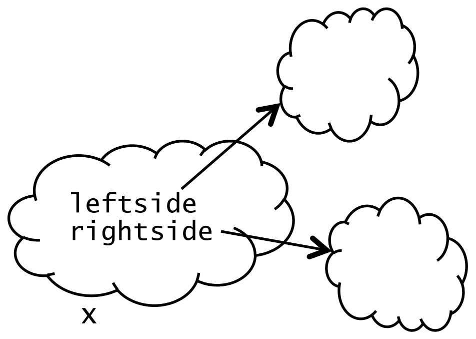
```
#include "Mystery.h"

int main() {
    Mystery x;
    Mystery y;

    // point A
    x = y;
    // point B

    return (0);
}
```

(a) (2 marks). The figure below shows the two objects `x` and `y` at point `A` in the code. Adjust the diagram to show the two objects at point `B` in the code.



(b) (6 marks). There are one or more problems with the code that result from the execution in `main`. What single member method would you add to the class `Mystery` to address these problem(s)? Write below the implementation of the method you add.

This Page is Intentionally Left Blank – Use for Rough Work