University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2017

Midterm Test

Examiners: T.S. Abdelrahman and D. Yuan

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. (8 marks). *General.*

Answer each of the following questions by circling either **Yes** or **No**.

Yes or **No**? When you allocate memory using the following statement:

```
    int *p = new int[5];
```

then you free the memory using:

```
    delete p;
```

Yes or **No**? If you put the implementation of a function into the header file, the compiler will give you an error.

Yes or **No**? It is safe for a function to return a pointer to an object that was created on the stack inside the function.

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

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

Yes or **No**? The compiler will generate a copy constructor for each class if you do not provide one.

Yes or **No**? You must write an overloaded "`operator=`" function for every class you create if you wish to use the assignment operator with your objects (e.g., "`a = b;`").

Yes or **No**? If a program has a bug, it will always show when running the program through the debugger.

Question 2. (4 marks). *Classes and Objects.*

Consider the following program.

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

      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() const {
       cout << “(“ << x << “,” << y  << “)” << endl;
}


int main() {
    Point a(2,3);
    // Evaluation is done left to right
    a.increment_x().increment_y().print();
    a.print();
    return 0;
}
```

Assuming the C++ compiler does not optimize away copying of objects. Write the output produced by the program.

Output:

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

(a) A pointer variable cannot point to other pointer variables. Circle on answer.

True **False**

(b) A pointer can be used to access elements of an array. Circle on answer.

True **False**

(c) Which of the following statements initializes the pointer `ptr` to the address of the variable `x`? Circle one answer.

`int* ptr=x;` `int* ptr=&x;` `int& ptr=&x;` `int* ptr=*x;`

(d) What happens in this segment of code? Circle one answer.

```
int a = 100;
int b = 200;
int* p = &a;
int* q = &b;
p = q;
```

a. `b` is assigned to `a`

b. `p` now points to `b`

c. `a` is assigned to `b`

d. `q` now points to `a`

(e) The correct prototype of a function `fun` that takes pointer to a `float`, a pointer to a pointer to a `char` and returns a pointer to a pointer to an `int` is (circle one answer):

```
int **fun(float**, char**);

int *fun(float*, char*);

int **fun(float*, char**);

int ***fun(*float, **char);
```

(f) Given the following declarations, what is the value of each of the following expressions? Write your answer where indicated.

```
    int i=10;
    int *pi=&i;
    double d=12.5;
    double *pd=&d;
```

```
i+1;
```
value:

```
(*pi) + 1;
```
value:

```
(*pd) + 1;
```
value:

(g) What is the value printed from the following C++ program segment? Write your answer down, where indicated.

```
int i=10;
int *p;
int **q;
int ***r;
p=&i;
*p=15;
q=&p;
**q=20;
r=&q;
***r=(*p) + 1;
cout << i;
```

value:

Question 4. (10 marks). *C++ I/O.*

Write a C++ function `void Parser()` that repeatedly reads floating point numbers from the standard input (using `cin`) and then immediately outputs the input numbers (using `cout`), one number per line. The function is to return when either: (1) a non-floating point number is input; or, (2) the end-of-file is reached. In the former case, "Invalid input" should be output on a line by itself before the function returns. In the latter case "End of File reached" should be output on a line by itself before the function returns.

Write your code in the box below.

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

void Parser() {

}
```

Question 5. (4 marks). *Parameter Passing.*

The `main` function of a program declares two variables, `array_size` and `array_ptr` as follows:

```
int  array_size;
int* array_ptr;
```

The `main` function invokes a function called `allocate_array` that has a <u>return type of `void`</u> and takes <u>two arguments</u>. The function allocates an array of integers that has `array_size` elements and makes `array_ptr` point to it. Which of the following are correct prototype and invocation of this function? Circle all answers that apply.

a. Function prototype: `void allocate_array (int n, int*& p);`
   Invocation in main: `allocate_array(array_size, array_ptr);`

b. Function prototype: `void allocate_array (int n, int*& p);`
   Invocation in main: `allocate_array(array_size, *array_ptr);`

c. Function prototype: `void allocate_array (int n, int*& p);`
   Invocation in main: `allocate_array(array_size, &array_ptr);`

d. Function prototype: `void allocate_array (int n, int** p);`
   Invocation in main: `allocate_array(array_size, &array_ptr);`

e. Function prototype: `void allocate_array (int n, int* p);`
   Invocation in main: `allocate_array(array_size, array_ptr);`

f. Function prototype: `void allocate_array (int n, int** p);`
   Invocation in main: `allocate_array(array_size, *array_ptr);`

Question 6. (4 marks). *Parameter Passing and Arrays.*

Consider the following function, called `max2`.

```
void max2 (int a[], int size, int & m1, int m2) {
  m1 = m2 = -1;
  for (int i = 0; i < size; ++i) {
    if (a[i] > m1) {
      m2 = m1;
      m1 = a[i];
    } else if (a[i] > m2) {
      m2 = a[i];
    }
  }
}
```

The array `a[5]` has the following elements: `{3,5,4,10,30}`. The `max2` function is invoked as follows: `max2 (a, 5, em1, em2)`, where `em1` and `em2` are integers both assigned to 0 before the invocation. What are the value of `em1` and `em2` after the function returns? Circle one answer.

(a) em1 is 0 and em2 is 0

(b) em1 is -1 and em2 is -1

(c) em1 is -1 and em2 is 30

(d) em1 is 30 and em2 is 10

(e) em1 is 10 and em2 is 30

(f) em1 is 30 and em2 is 0

(g) em1 is 3 and em2 is 4

(h) em1 is 4 and em2 is 3

(i) None of the above

Question 7. (9 marks). *Class Method Design.*

Sets are a common abstraction in mathematics, science and engineering. The following is a class definition for sets of integers. Assume it is in a file called `set.h`.

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

#define MAX_SIZE 10

class set {
  private:
        int size;                // number of elements in the set
        int elements[MAX_SIZE];  // elements of the set

   public:
        // creates an empty set
        set();

        // returns true if e is in set
        bool isMember(int e) const;

        // returns the size of the set
        int getSize() const;

        // returns true if the set is empty
        bool isEmpty();

        // adds element e to the set, returns false if element
        // if already exists in set, otherwise returns true
        bool add(int e);

        // returns a new set that is the union of two sets
        set operator+ (const set & rhs) const;

        // more functions that are not relevant to the question
        :
};
```

(a) (3 marks). Write the implementation of the class member function `isMember`, appearing in a file called `set.cpp`. Use the space below. Your answer should be a few lines of code.

```
                                                    {      ← Write function header here
---------------------------------------------------------
                                                           ← Write function body here

 }
```

The overloaded "+" operator for the `set` class allows code like this to be written:

```
set X;
X.add(4);
X.add(5);

set Y;
Y.add(5);
Y.add(6);
   :
set Z;
Z = X + Y;
```

The result of `X + Y` is a new set that is the union of the sets `X` and `Y`. That is, the new set has the elements 4, 5 and 6. Note that the common element 5 appears only once in the resulting set.

(b) (6 marks). Write the implementation of the overloaded `operator+` function, as a member of `set`. Clearly show the function header and its body. Again, the function is written in the file `set.cpp`. Write your answer in the box below. You may want to use the functions `isMember` and/or `add`.

```
                                                   {     ← Write function header here

                                                         ← Write function body here

}
```

Question 8. (10 marks). *Constructors and Destructors.*

Consider the following definition of the class `Complex`.

```
class complex {
  private:
        float real;
        float imag;
   public:
        complex();
        complex(float r, float i); // Two-floats constructor
        complex& addComplex (complex other);
        complex& operator= (complex rhs);
};
```

The following is the implementation of the class methods.

```
complex::complex() {
      real=0.0;
      imag=0.0;
}

complex::complex(float r, float i) {
      real=r;
      imag=i;
}

complex& complex::addComplex (complex other) {
     complex tmp(1.0,1.0);
     real = real + other.real + tmp.real;
     imag = imag + other.imag + tmp.imag;
     return (*this);
}

complex& complex::operator= (complex rhs) {
     real = rhs.real;
     imag = rhs.imag;
     return (*this);
}
```

Assume the class definition and implementation to be correct.

Consider the execution of the following `main` function, from the time `main` is invoked until (and immediately after) it returns.

```
int main() {
    complex a;
    complex b(1.0,2.0);
    complex c;
    complex d[2];

    c = a.addComplex(b);

    return (0);
}
```

Assuming the C++ compiler does not optimize away copying of objects.

(a) (2 marks). How many times is the default constructor invoked? Circle one answer.

1. 2 times
2. 3 times
3. 4 times
4. 5 times
5. None of the above

(b) (2 marks). How many times is the copy constructor invoked? Circle one answer.

1. 0 times
2. 1 time
3. 2 times
4. 3 times
5. None of the above

(c) (2 marks). How many times is the two-floats constructor invoked? Circle one answer.

1. 0 times
2. 1 time
3. 2 times
4. 3 times
5. None of the above

(d) (2 marks). How many times is the destructor invoked? Circle one answer.

1. 0 times
2. 4 times
3. 3 times
4. 8 times
5. None of the above

(e) (1 mark). How many times is the overloaded assignment operator invoked? Circle one answer.

1. 0 times
2. 1 time
3. 2 times
4. 4 times
5. None of the above

(f) (1 mark). How many times is the overloaded addition operator invoked? Circle one answer.

1. 0 times
2. 2 times
3. 4 time
4. 3 times
5. None of the above

Question 9. (8 marks). *Dynamic Memory Allocation.*

Given the following program, indicate the number of object of type Foo that exist in memory when the program execution reaches:

Point X:

Point Y:

Point Z:

Point W:

```
void do_something_else(Foo obj) {
   Foo* ptr = new Foo();
   // Point Y
   return;
}

void do_something (Foo* arg) {
    Foo a;
    Foo b;
    do_something_else(*arg);
    Foo* ptr = new Foo();
    delete arg;
    // Point Z
    return;
}

int main() {
    Foo a;
    Foo* p;
    // Point X
    p = new Foo();
    do_something(p);
    // Point W
    return (0);
}
```

Question 10. (9 marks). *Arrays and Objects.*

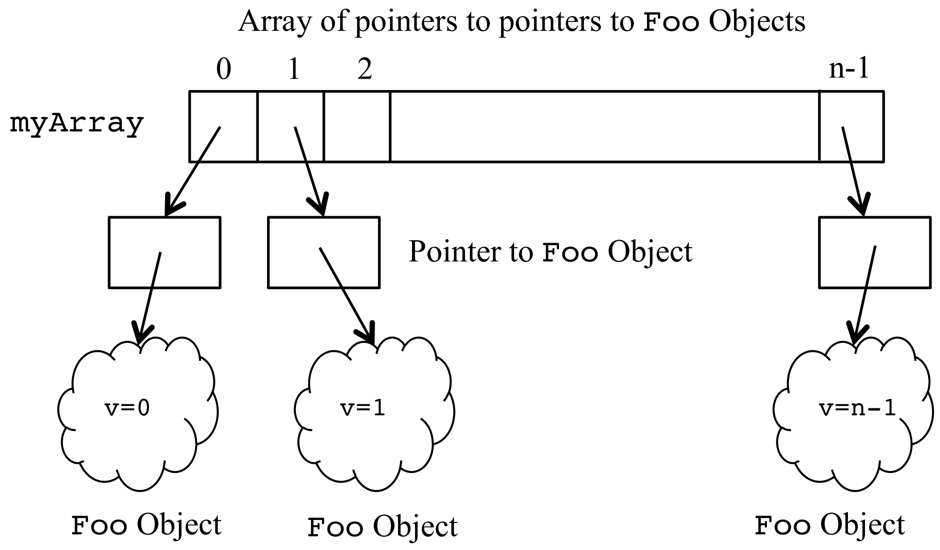
Consider the following definition and implementation of the class `Foo`. Assume it is correct.

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

class Foo {
   private:
       int value;
   public:
           Foo(int v);
          ~Foo();
       int  getValue() const;
       void print() const;
};

Foo::Foo(int v) {value = v;}
Foo::~Foo() { }
int Foo::getValue() const { return v;}
void Foo::print() const {cout << v << endl;}
```

We wish to allocate the data structure shown below, *dynamically* at run time. 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, called `myArray`. Each element of the array points to a dynamically allocated pointer that can point to object of type `Foo`. Each of these dynamically allocated pointers points to a dynamically allocated `Foo` object.



Assuming an integer variable `n` whose value is read from the standard input using `cin`, as shown below.

```
int n;
cin >> n;
```

(a) **(1 marks)**. Write the declaration of the variable `myArray`.

(b) **(1 marks)**. Write a C++ code segment to dynamically allocate the array `myArray` of size `n`.

(c) **(2 mark)**. Write a C++ code segment to dynamically allocate the `n` pointers and assign the elements of the array `myArray`, as shown in the figure above.

(d) **(3 mark)**. Write a C++ code segment to dynamically allocate `n` Foo objects and have their addresses stored in the `n` dynamically allocated pointers, as shown in the figure above. The value of the private data member of each object should be initialized as shown in the figure.

(e) **(2 marks)**. Write a C++ code segment to de-allocate all the dynamically allocated data in the parts above so that no memory leaks exist.

Question 11. (5 marks). *Constructors and Destructor.*

Consider the following definition/implementation of a class called `Square`.

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

class Square {
  private:
    int length;
    int width;
  public:
    Square() {
      cout << "Constructor 1" << endl;
      length = 0;
      width = 0;
    }

    Square(int _length, int _width) {
      cout << "Constructor 2" << endl;
      length = _length;
      width = _width;
    }

    Square(const Square& s) {
      cout << "Constructor 3" << endl;
      length = s.length;
      width = s.width;
    }

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

    int get_length() const { return length; }

    int get_width() const { return width; }
};
```

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

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

Square g(1,2);

int size (Square s) {return s.get_length()*s.get_width();}
int size_r (Square &s) {return s.get_length()*s.get_width();}

int main () {
  Square a;
  Square &r = a;
  Square *p;
  p = new Square(3,4);
  Square *q = p;
  int size1 = size (*q);
  delete q;
  int size2 = size_r (r);
  cout << "size of a: " << size1 << "; size of r: " << size2 << endl;
  return 0;
}
```

What is the output of the program?

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

A	Constructor 2 Constructor 2 Constructor 1 Constructor 3 Destructor Destructor size of a: 6; size of r: 2 Destructor Destructor	E	Constructor 1 Constructor 2 Constructor 2 Constructor 3 Destructor Destructor size of a: 12; size of r: 0 Destructor Destructor
B	Constructor 2 Constructor 1 Constructor 2 Constructor 3 Destructor Destructor size of a: 12; size of r: 0 Destructor Destructor	F	Constructor 2 Constructor 1 Constructor 2 Destructor size of a: 12; size of r: 0 Destructor Destructor

C	Constructor 1 Constructor 2 Constructor 3 Destructor Destructor size of a: 12; size of r: 0	G	Constructor 2 Constructor 1 Constructor 2 Constructor 3 Destructor Destructor size of a: 6; size of r: 0 Destructor Destructor
D	Constructor 2 Constructor 1 Constructor 2 Constructor 3 Destructor Destructor size of a: 2; size of r: 0 Destructor Destructor	H	None of the above

Question 12. (5 marks). *Classes and Objects.*

The following class definition describes a simple C++ class called `mClass`. The definition is in a file called `mClass.h`. Assume this definition to be correct.

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

class mClass {
private:
        int m;
public:
        mClass(int v);
        mClass(const mClass& src);
       ~mClass();
        int getM() const;
        void setM(int new_m);
        mClass& addM(const mClass & other);
};
```

The implementation of the class is in the file `mClass.cpp`, shown below. The line number at the beginning of each line is **not** part of the code; it is there only to facilitate your answer.

```
01: #include “mClass.h”

02: mClass::mClass(int v) {m = v;}

03: mClass::mClass(const mClass& src) {m = src.m;}

04: mClass::~mClass() { }

05: int mClass::getM() const {
06:     m = m + 1;
07:     return m;
08: }

09: void mClass::setM(int new_m){
10:     m = new_m;
11: }

12: void mClass::addM(const mClass& other) {
13:     m = other.m;
14:     other.m = m + 1;
15: }
```

Now consider the following code, which uses `mClass`:

```
    #include <iostream>
    using namespace std;
    #include "mClass.h"
    int main() {
16:     mClass a(5);
17:     cout << a.m << endl;
18:     mClass* p = new mClass(8);
19:     a.addM(p);
20:     cout << p->getM() << endl;
21:     mClass c;
22:     return (0);
23: }
```

The code is compiled using the command: `g++ main.cpp mClass.cpp —o main.exe`. However, there are several compiler errors that result. In the table below, indicate for each line number (shown in the code above) whether an error occurs or not. If there is no error, leave the error description blank. Otherwise, write a one short sentence description of the error.

Line Number	Error (leave blank or error description)
01	
02	
03	
04	
05	
06	
07	
08	

09	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	

Question 13. (4 marks). *Classes and Objects.*

The C++ class `aClass` is shown below. Assume it is correct.

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

class aClass {
private:
        int* a;
public:
        aClass(int v);
       ~aClass();
        void setA(int v);
        void print() const;
};

aClass::aClass(int v) {
     a = new int;
     *a = v;
}

aClass::~aClass() { delete a; }
void aClass::setA(int v) {*a = v;}
void aClass::print() const { cout << *a << endl;}
```

The following `main` function uses this class:

```
int main() {
     aClass x(3);
     aClass y(5);
     x.print();     // First print
     y.print();     // Second print
     x = y;
     y.setA(7);
     x.print();     // Third print
     y.print();     // Fourth print
}
```

Indicate in the table below the values printed by each of the `print` function invocations above.

First print	
Second print	
Third print	
Fourth print	

Question 14. (10 marks). *Compilation.*

(a) (3 marks). Consider the following program:

```
#include <iostream>
#define A 3
#define int_var A

using namespace std;
int main () {
  int_var = int_var + 1;
  cout << int_var << endl;
  return 0;
}
```

What is the output produced by the program? Circle one answer.

(a) None, the program has compilation error(s)
(b) 3
(c) 4
(d) 0
(e) None of the above

(b) (3 marks). Consider the following program:

```
#include <iostream>
#define A 3
#define int_var a

using namespace std;
int main () {
  int a = 0;
  a++;
  int_var = int_var + A;
  cout << a << endl;
  return 0;
}
```

What is the output produce by the program? Circle one answer.

(a) None, it has compilation error(s)
(b) 3
(c) 4
(d) 1
(e) 0
(f) None of the above

(c) **(4 marks)**. 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`.

Write down the Unix commands necessary to separately compile the above files and generate the executable.

Write your answer here