University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2016

Final Examination

Examiners: T.S. Abdelrahman and D. Yuan

Duration: Two and a Half Hours

This exam 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 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. _____	**Q16.** _____

Total: ☐

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

(a) Consider the following recursive function:

```
int Fun (int n) {
    if (n == 4) return (2);
    else return (2* Fun(n+1));
}
```

What is the value returned by the function call `Fun(2)`?

(b) **Yes** or **No**? Every class has at least one constructor function, even when none is declared.

(c) **Yes** or **No**? Calling a virtual function is slower than calling a non-virtual function.

(d) **Yes** or **No**? A derived class constructor always calls the default base class constructor to initialize and allocate memory for the base object.

(e) **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 (i.e., "a = b;").

(f) **Yes** or **No**? The following C++ statement is incorrect because the function returns an integer but has a `void` return type: `virtual void foo( int, float ) = 0 ;`

(g) **Yes** or **No**? The worst-case complexity of quick sort is $O(n^2)$.

(h) **Yes** or **No**? Each non-leaf node in a binary search tree always has exactly 2 children.

(i) **Yes** or **No**? The worst-case complexity of search in any binary search tree is O(log(n)).

(j) **Yes** or **No**? A node in a binary tree has at most one parent node.

(k) **Yes** or **No**? A derived class inherits a private member variable in the base class.

Question 2. (4 marks). *Pointers and Parameter Passing.*

A function `updatePointer` takes one argument, of type `int*`, and changes the value of this argument. Assume the formal argument of `updatePointer` is a variable called `ptr` and that the function is called with an actual argument called `my_ptr`. Thus, the function changes the value of `my_ptr`. Which of the following are the correct prototype and invocation of the function?

Circle **A**, **B**, **C**, etc. to indicate which are correct. Circle <u>all</u> that are correct.

A:

function prototype: `void updatePointer (int* ptr);`
invocation (i.e., function use): `updatePointer (my_ptr);`

B:

function prototype: `void updatePointer (int&* ptr);`
invocation (function use): `updatePointer (&my_ptr);`

C:

function prototype: `void updatePointer (int*& ptr);`
invocation (i.e., function use): `updatePointer (my_ptr);`

D:

function prototype: `void updatePointer (int*& ptr);`
invocation (function use): `updatePointer (&my_ptr);`

E:

function prototype: `void updatePointer (int*& ptr);`
invocation (i.e., function use): `updatePointer (*my_ptr);`

F:

function prototype: `void updatePointer (int** ptr);`
invocation (function use): `updatePointer (&my_ptr);`

Question 3. (4 marks). *Functions.*

Consider the following code of a class `Pair`. You may assume that the code is error free.

```
class Pair {
  private:
     int x;
     int y;
  public:
     Pair(int x_value, int y_value);
     int getX() const;
     int getY() const;
     void setX(int x_value);
     void setY(int y_value);

};

Pair::Pair(int x_value, int y_value) {
    x = x_value;
    y = y_value;
}

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

void Pair::setX(int x_value) {x = x_value;}

int Pair::getY() const {return y;}

void Pair::setY(int y_value) {y = y_value;}
```

We wish to be able to do the following in a non-member function, like `main`:

```
Pair mypair(10,20);
cout << mypair << endl;
```

to print: `(10,20)`.

(a) (1 mark). What operator must be overloaded to make the above operation correct? Only write the name of the operator.

(b) (3 marks). Write the implementation of this operator. Note that you cannot change the definition or implementation to the class `Pair`, except possibly to add function prototypes to it.

Question 4. (12 marks). *Pointers and Linked Lists*.

In this question, you need to complete the implementation of an ordered linked list using `struct`. The skeleton of the code is provided for you below. You need to implement the "insert" function. It should insert a new node pointed by the parameter `new_node`. Assume that the memory space for the new node *has already been allocated by the caller* (so that insert does not need to allocate memory space for the new node). In addition, you should keep the linked list sorted from the smallest value to the largest. For example, if the user inserts three nodes: "3", "187", "5" in this order, your insert function should make sure the linked list be in the order: 3 -> 5 -> 187. Assume that there will be no duplicate values.

An example of main function is provided to showcase how one uses the insert function. You do not need to worry about freeing unused memory blocks in this question.

```
struct node {
  int data;
  struct node *next;
};


struct node *head = NULL;

void insert(struct node *new_node) {
  /* This first line is correct and is given to you
  struct node **link = &head;

  /* The code snippet of your choice goes here. */
}

int main() {
  struct node *p = new node;
  p->data = 3;
  insert(p);
  p = new node;
  p->data = 187;
  insert(p);
  p = new node;
  p->data = 5;
  insert(p);
  return 0;
}
```

The insert function body consists of only 5 lines of code. The first line is already given to you. For each of the remaining four lines, you are to choose from the list below of possible code lines so that they form the correct implementation for insert. For example, if you choose A, C, I, J, your insert function will look like:

```
void insert(struct node *new_node) {
  struct node **link = &head;
  /* The code snippet of your choice goes here. */
  while (link != NULL && link->data < new_node->data) // A
        link = &((*link)->next); // C
  new_node->next = ***link; // I
  *link = &new_node; // J
}
```

Please write down your choice. Note that there is **one** correct combination. If you feel there is more than one, **only choose one combination**.

Line 1: [ ] (write one choice, from A to J)

```
(A)  while (link != NULL && link->data < new_node->data)
(B)  while (link != NULL && (*link)->data < new_node->data)
(C)  while (*link != NULL && (*link)->data < new_node->data)
(D)  while (*link != NULL && (*link).data < new_node->data)
(E)  while (**link != NULL && (**link)->data < new_node->data)
(F)  while (link != NULL && link->data > new_node->data)
(G)  while (link != NULL && (*link)->data > new_node->data)
(H)  while (*link != NULL && (*link)->data > new_node->data)
(I)  while (*link != NULL && (*link).data > new_node->data)
(J)  while (**link != NULL && (**link)->data > new_node->data)
```

Line 2 : [ ] (write one choice, from A to E)

```
(A)    link = link->next;
(B)    link = (*link)->next;
(C)    link = &((*link)->next);
(D)    link = (**link)->next;
(E)    *link = (*link)->next;
```

Line 3: [ ] (write one choice, from A to J)

```
(A)  *link = new_node;
(B)  link = new_node;
(C)  link = &new_node;
(D)  **link = *new_node;
(E)  *link = &new_node;
(F)  new_node->next = *link;
(G)  new_node->next = &link;
(H)  new_node->next = **link;
(I)  new_node->next = ***link;
(J)  *(new_node->next) = **link;
```

Line 4: [ ] (write one choice, from A to J)

```
(A)  new_node->next = *link;
(B)  new_node->next = &link;
(C)  new_node->next = **link;
(D)  new_node->next = ***link;
(E)  *(new_node->next) = **link;
(F)  *link = new_node;
(G)  link = new_node;
(H)  link = &new_node;
(I)  **link = *new_node;
(J)  *link = &new_node;
```

Hint: You should use the space below to draw a picture of the linked list and what the various pointers are pointing to.

Question 5. (6 marks). *Class Definition.*

Consider the following definition of the class `Complex` and its use in the `main` function:

```
class Complex {
   private:
      float real;
      float imag;
   public:
      float getReal();
      float getImag();
      void setReal(float r);
      void setImag(float i);
      void print();
};

int main() {

    Complex a;

    a.setReal(1.0);
    a.setImag(-9.6);

    Complex b(a);
    Complex c;

    c = a;
    a.print();
    b.print();
    c.print();
    return(0);
}
```

A programmer decides to change the class definition into the following:

```
struct _complex {float real; float imag;};

class Complex {
   private:
      struct _complex* number;
   public:
      // May need to add new function members or delete some
      float getReal();
      float getImag();
      void setReal(float r);
      void setImag(float i);
      void print();
};
```

However, the programmer must ensure that nothing needs to change in the `main` function, so users of the `Complex` class are unaware of the change. With this in mind, answer the following questions.

(a) **(2 marks)**. What additional (i.e., new) class member functions **must** be defined and implemented 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**.

Question 6. (6 marks). *Objects.*

Study the following class definition (`point.h`) and implementation (`point.cc`).

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

class point {
  private:
    bool set;
    struct coord* thepoint;

  public:
     point();
     point(int xv, int yv);
     ~point();
     void setX(int xv);
     void setY (int yv);
     point operator+ (point rhs);
     void print();
};
```

```
#include "point.h"
#include "iostream"
using namespace std;

point::point() {
     set = true;
     thepoint = new struct coord;
     thepoint->x = 0;
     thepoint->y = 0;
}

point::point(int xv, int yv) {
     set = true;
     thepoint = new struct coord;
     thepoint->x = xv;
     thepoint->y = yv;
}

point::~point() {
}

void point::setX(int xv) {
     thepoint->x = xv;
}


void point::setY(int yv) {
     thepoint->y = yv;
}
```

```
point point::operator+ (point other) {
     point sum;
     sum.thepoint->x = thepoint->x + other.thepoint->x;
     sum.thepoint->y = thepoint->y + other.thepoint->y;
     return (sum);
};

void point::print() {
     cout << "(" << thepoint->x
          << "," << thepoint->y
          << ")" << endl;
}
```

Now consider the following main function, which uses the above class.

```
#include "iostream"
#include "point.h"
using namespace std;

int main () {
    point a(1,1);
    point b(5,12);
    point c(b);
    point d;

    a.print(); // Statement # 1
    b.print(); // Statement # 2
    c.print(); // Statement # 3
    d.print(); // Statement # 4

    a.setX(0);
    b.setX(8);
    c.setY(20);
    d.setX(5);
    d.setY(10);

    a.print(); // Statement # 5
    b.print(); // Statement # 6
    c.print(); // Statement # 7
    d.print(); // Statement # 8

    c=a;
    d=a+b;


    a.print(); // Statement # 9
    b.print(); // Statement # 10
    c.print(); // Statement # 11
    d.print(); // Statement # 12

    return (0);

}
```

Indicate what each statement in the above main function prints. For simplicity, each statement that produces output has been given a number, and you can write the output of each statement in the table below.

Statement #	Output
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Hint: Draw a picture!

Question 7. (2 marks). *Lab Assignments.*

In lab assignment 5, you were asked to implement a binary search tree consisting of objects of type `treeNode`. The class `treeNode` has pointers to the left and right subtrees, such as:

```
class treeNode {
    public:
       treeNode* left;
       treeNode* right;
      :
};
```

Assume that your program creates a binary search tree. A global variable called `myroot` points to the root of this tree. We wish that when the following code is executed, the entire tree is deleted, i.e., all its nodes are deleted:

```
delete myroot;
```

Write the destructor of `treeNode` so that the above statement deletes all the nodes in the tree.

Provide your answer below. You answer should be very short. Long answers will be penalized!

```
treeNode::~treeNode() {




}
```

Question 8. (2 marks). *Inheritance.*

What output does the following program print?

```
class Base {
public:
    virtual void display();
};

void Base::display() {
    cout << "In base class \n";
}

class Derived : public Base {
public:
    Derived(int v );
    virtual void display();
private:
    int value;
};

Derived::Derived(int v) {
    value = v;
}

void Derived::display() {
    cout << "In derived class, value is " << value << "\n";
}

int main()
{
    Base b;
    Derived d(4);
    b = d;
    b.display();
    Base* bp = new Derived(7);
    bp->display();
    return(0);
}
```

Output:

Question 9. (7 marks). *Inheritance.*

Consider the following definitions for the `Vehicle` and the `Car` classes.

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

class Vehicle {
    private:
        int v_id;
        char *model;
    public:
        Vehicle();
        Vehicle(int id, char *m);
        ~Vehicle();
        int get_id();
        virtual void move(int _x, int _y);
};


Vehicle::Vehicle() {
    v_id = 0;
    model = new char [8];
    strcpy(model, “unknown”);
    cout << "Vehicle: constructor: " << v_id << endl;
}

Vehicle::Vehicle(int id, char *m) {
    v_id = id;
    model = new char [strlen(m) + 1];
    strcpy(model, m);
    cout << "Vehicle: constructor: " << v_id << endl;
}

Vehicle::~Vehicle() {
    move(0, 0);
    model = NULL;
}


int Vehicle::get_id() {
    move(0, 0);
    return v_id;
}


void Vehicle::move(int _x, int _y) {
    cout << "Vehicle: move: " << v_id << " " << model << endl;
}
```

```
class Car : public Vehicle {
    private:
           int x, y;
           void print_position(void);
    public:
           Car(int id, char *m, int _x, int _y);
           ~Car();
           int get_id();
           virtual void move(int _x, int _y);
};

Car::Car(int id, char *m, int _x, int _y) : Vehicle(id, m) {
    x = _x; y = _y;
}

Car::~Car() {
    print_position();
}

int Car::get_id() {
    print_position();
    return Vehicle::get_id();
}

void Car::move(int _x, int _y) {
    x += _x; y += _y;
    cout << "Car: move: x = " << x << ", y = " << y << endl;
    Vehicle::move(_x, _y);
}

void Car::print_position(void) {
    cout << "Car: position: x = " << x << ", y = " << y << endl;
}


    // main program
    // Line numbers are to facilitate answers, not part of code
    int main() {
 1     Vehicle v;
 2     Vehicle *vp = &v;
 3     Car c(10, "Zip", 2, 6);
 4     Car *cp = &c;
 5     cout << vp->get_id() << endl;
 6     vp->move(1, 3);
 7     cp->move(2, 1);
 8     *vp = *cp;
 9     vp->move(1, 1);
10     vp = cp;
11     cout << vp->get_id() << endl;
12     cp->move(1, 1);
13     cout << “Done ..” << endl;
14  }
```

Show the output produced by each line of the program in the table below. If a line has no output, write N/A. Refer to the line numbers in `main()` above.

Line #	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

Question 10. (4 marks). *Operator Overloading.*

Assume you are given the definition of class `A` below and assume that this code compiles correctly. Moreover, assume that an implementation of class `A` exists in the form of an `A.o` file, but you do not have access to the source files.

```
class A {
  private:
    Poodle *bp ;
  public:
    A();
    ~A();
    A( const A& a ) ;
    A& operator=( const A& rhs ) ;
} ;
```

Further, assume you are given the definition of class `B` below as well as an implementation of the copy constructor for `B` that works correctly:

```
class B: public A {
  private:
    Poodle *pp ;
  public:
    B();
    ~B();
    B( const B& b ) ;
    B& operator=( const B& rhs ) ;
} ;

B::B(const B& b):A(b) {
    pp = new Poodle(*(b.pp)) ;
}
```

Based on the code you have available, provide the implementation of the overloaded assignment operator for the class `B`.

```
B& operator=(const B& rhs) {




}
```

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

Consider the C++ code shown below for a base class `onePoint`. The implementation is correct.

```
class onePoint {
    private:
        int x;
    public:
        bool valid;
        onePoint(int xv);
        ~onePoint();
        void scale();
        virtual void reflect();
        virtual void print()=0;
};

onePoint::onePoint(int xv) {
    valid = true;
    x = xv;
}

onePoint::~onePoint() {
    // Nothing to do
}

void onePoint::scale() {
    x = 2*x;
}

void onePoint::reflect() {
    x = -1*x;
}

void onePoint::print(){
    cout << "x = " << x << " ";
}
```

Now consider the following class, `towPoints`, which derives from `onePoint`.

```
class towPoints : public onePoint {
    private:
        int y;
    public:
        towPoints(int xv, int yv);
        ~towPoints();
        void scale();
        virtual void reflect();
        virtual void print();
};


towPoints::towPoints(int xv, int yv):onePoint(xv) {
    y = yv;
}
```

```
towPoints::~towPoints() {
    // Nothing to do
}

void towPoints::scale() {
    /* Deliberately left empty */
}

void towPoints::reflect() {
    /* Deliberately left empty */
}

void towPoints::print(){
    onePoint::print();
    cout << “, y = “ << y << endl;
}
```

Based on the above definitions and implementations of the two classes, indicate by placing an X in the appropriate column whether each of the following code segments is *correct* code or *incorrect* code. A segment of code is correct if it produces no compile-time error. You should assume each part of the questions to be independent of the others.

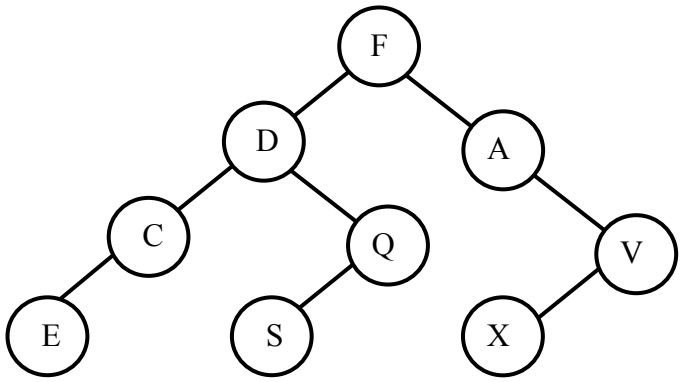
Code Segment	Correct	Incorrect
`// In main()` `onePoint p0;`		
`// In main()` `onePoint p1(2);`		
`// In main()` `towPoints p2(3,5);`		

`void towPoints::reflect() {`
`  x = -1*x;`
`  y = -1*y;`
`}`		

`towPoints p3(10,10);` `p3.valid = false;`		
`towPoints p4(5,7);` `p4.x = 4;`		

Question 12. (3 marks). *Tree Traversals*.

Give the inorder, preorder, and postorder traversals of the tree shown below.



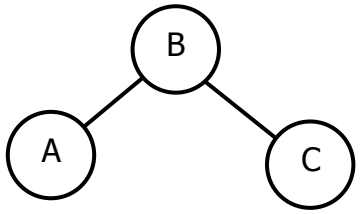
Inorder Traversal:

Preorder Traversal:

Postorder Traversal:

Question 13. (8 marks). *Tree Traversals*.

(a) (4 marks). In the ***reverse inorder*** traversal of a tree, the right subtree of each node is first traversed (recursively in reverse inorder), the node then is visited, and finally the left subtree of the node is traversed (recursively in reverse inorder). That is, the order of the traversal is RNL. For example, the reverse inorder traversal of the tree shown below is C B A.



Write a ***recursive*** function to perform the reverse-inorder traversal of a binary tree. Assume that visiting a node simply prints its key to `cout`. **Your code should be very short (4-6 lines)! Long code will be penalized**!

You may assume the following declarations:

```
class treenode {
      public:
         int data;
         treenode *left;
         treenode *right;
};
treenode *Root; // root of the tree


void reverseorder (treenode *rt) {

}


// This is how reverseorder is called
reverseorder(Root);
```

(b) (4 marks). A tree ***T*** has the following inorder and preorder traversals:

```
Preorder traversal:   1 8 12 25 13  7 9
Inorder  traversal:   8 1 25 12  7 13 9
```

Draw **the** tree ***T***. Please note that there is only one tree ***T*** that has the inorder and preorder traversals shown above. Do **NOT** draw two trees; draw only one tree whose inorder and preorder traversal are as shown above.

Question 14. (6 marks). *Binary Search Trees.*

The following is the code for the insert function for a binary search tree (BST), similar to what you have written for lab assignment 5.

```
class TreeNode {
private:
  int value;
  TreeNode *left;
  TreeNode *right;
public:
       TreeNode(int v);
  void insert (int v);
  :
};


void TreeNode::insert (int v) {
  if (value == v)
    return;
  if (v < value) {
     if (left == NULL)
        left = new TreeNode(v);
     else
        left->insert(v); // recursion
  }  else {
     if (right == NULL)
        right = new TreeNode(v);
    else
        right->insert(v); // recursion
  }
  return;
}
```

Draw the BST that results from inserting nodes with the `value`s 1 to 7 in the following order (from left to right):

5 3 7 6 2 1 4

1 2 3 4 5 6 7

4 3 5 2 6 1 7

Question 15. (12 marks). *Complexity Analysis.*

Determine the ***worst-case*** time complexity (expressed in big-O notation) for each of the program segments below as a function of the size of the input `n`. Show the details of your analysis and clearly indicate your final result.

(a) (2 marks) The size of the input is `n`.

```
w=0;
for (int i=0; i < n; ++i) {
  for (int j=0; j < n*n; ++j) {
    for (int k=0; k < n*n*n; ++k) {
       w = w + 1;
    }
  }
}
```

T(n) = O ( )

(b) (2 marks). The size of the input is `n`.

```
for (int i=0; i < n; ++i) {
       for (int j=0; j < i*n ; ++j) {
         O(1)
       }
}
```

T(n) = O ( )

(c) (2 marks). Give the time complexity of the following segment of code.

```
for(int i=0; i<10; i++)
   for(int j=0; j<n; j++)
      for(int k=n-2; k<n+2; k++)
         cout << i << " " << j << endl;
```

T(n) = O ( )

(d) (6 marks). Assume for simplicity that `n` is a power of two.

```
int mystery (int k, int n) {
   int x;
   int y;

   if (n <= 1) return(0);
   for (int i=0 ; i < n ; ++i) {
        O(1) // to determine the value of k
   }
   x = mystery (k, n/2);   // integer division
   y = mystery (n-k, n/2); // integer division
   return (x+y);
}
```

Write the recurrence equation for `T(n)`.

Solve the recurrence equation to obtain an expression of `T(n)` in terms of `n`.

Express `T(n)` using the big-O notation.

T(n) = O ( )

Question 16. (8 marks). *Complexity Analysis.*

Bubble sort is a basic sorting algorithm, which we did not cover in class, but its code is shown below.

```
void swap (int & x, int & y) {
  int tmp = x; x = y; y = tmp;
}

void bubbleSort(int* a, int length) {
  bool flag = true;

  for(int i = 0; (i < length) && flag; i++) {
     flag = false;
     for (int j = 0; j < (length - i - 1); j++) {
        if (a[j] > a[j + 1]) {
          swap (a[j], a[j+1]);
          flag = true;
        }
    }
  }
  return;
}
```

(a) (2 marks). What is the best-case complexity of `bubbleSort`, expressed in big-O notation?

T(n) = O( )

(b) (1 marks). Can you give an example of input array of length 5 that will result in this best-case scenario?

(c) (2 marks). What is the worst-case complexity of `bubbleSort`, expressed in big-O notation?

T(n) = O( )

(d) (1 marks). Can you give an example of input array of length 5 that will result in this worst-case scenario?

(e) (2 marks). What is the average-case complexity of `bubbleSort`, expressed in big-O notation?

T(n) = O( )