University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2011

Final Examination

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

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. _____	**Q10.** _____
Q2. _____	**Q11.** _____
Q3. _____	**Q12.** _____
Q4. _____	**Q13.** _____
Q5. _____	**Q14.** _____
Q6. _____	**Q15.** _____
Q7. _____	**Q16.** _____
Q8. _____	**Q17.** _____
Q9. _____	**Q18.** _____

Total | |

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

Answer the following questions either by **circling Yes** or **No**, or by providing a **very brief** and **direct** answer when such answer is required (none is required for **Yes** or **No** questions).

(a) **Yes** or **No**? A copy constructor is always called when an object is passed by value to a function.

(b) **Yes** or **No**? A static member function is a member function that can be invoked with or without an object.

(c) **Yes** or **No**? The only reason to pass parameters to a function by reference is allow the function to modify these parameters.

(d) **Yes** or **No**? If class C is derived from class B and class B is derived from class A, then if a class C object is being allocated, the constructors are called in the following order: constructor of class A first, constructor of class B second, and finally constructor of class C.

(e) **Yes** or **No**? If you destroy an object through a pointer to a base class, and the base-class destructor is not virtual, the derived-class destructors are not executed, and the destruction might not be complete.

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

(g) **Yes** or **No**? An `O(log(n))` algorithm is always faster than an `O(n)` algorithm.

(h) What is the worst-case time complexity for search in *any* binary search tree that has `n` nodes?

(i) In one sentence, explain why C++ does not make all member functions by default virtual.

(j) What is the time complexity of Quicksort if the `n` elements of the input array to be sorted are already sorted?

Question 2. (4 marks). *Objects with pointers.*

Study the following definitions and implementations of a class `ObjectName`, which is intended to store a character string that represents the name of some object.

```
class ObjectName {
   private:
      char* _thename;
      int length;
   public:
      ObjectName(const char* name);
      ~ObjectName();
       :
       :
};

ObjectName::ObjectName(const char* name) {
              length = strlen(name);
              _thename = new char[length+1];
              strcpy(_thename, name);
}

ObjectName::~ObjectName() {
              length = 0;
              _thename = NULL;
}
```

Modify the class functions so as no memory leaks exist. Write your modified code below.

Question 3. (7 marks). *Objects with pointers.*

Consider the following definitions (left) and partial implementations (right) of the classes `Wrapper`, `WrappedA` and `WrappedB`.

```
class Wrapper {
   private:
      WrappedA* a;
      WrappedB* b;
      int count;
   public:
      Wrapper();
      ~Wrapper();
       :
       :
};
```

```
Wrapper::Wrapper() {
        a = new WrappedA;
        b = new WrappedB[5];
        count = 6;
}

:
:
```

```
class WrappedA {
   private:
      WrappedB** x;
      int count;
   public:
      WrappedA();
      ~WrappedA();
       :
       :
};
```

```
WrappedA::WrappedA() {
         x = new WrappedB* [10];
         for (int i=0; i < 10; ++i)
              x[i] = new WrappedB;
         count = 10;
}

:
:
```

```
class WrappedB {
   private:
      int count;
   public:
      WrappedB();
      ~WrappedB();
       :
       :
};
```

```
WrappedB::WrappedB() {
           count = 1;
}

:
:
```

Write the destructor for each of the three classes so as no memory leaks exist. Provide your answer in the space below.

```
Wrapper::~Wrapper() {




}
```

```
WrappedA::~WrappedA() {




}
```

```
WrappedB::~WrappedB() {




}
```

Question 4. (9 marks). *Operator Overloading*.

Consider the definitions below for classes `Point` and `PointVector`.

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

class Point {
private:
   float x;
   float y;
public:
   Point (float _x, float _y) {x = _x; y = _y;}
   Point () {x = 0; y = 0; }
   float get_x () const { return x; }
   float get_y () const { return y; }
};


class PointVector {
private:
   int length;
   Point *points;
public:
   PointVector (int _length);
   Point operator [] (int index);
   friend void operator << (ostream & out,
                                    const PointVector & rhs);
};


PointVector::PointVector (int _length)
{
   length = _length;
   points = new Point[length];
}

int main () {
   PointVector pvec(3);
   Point p1 (3, 2);
   pvec[1] = p1;
   cout << pvec << endl;
}
```

(a) **(1 mark)**. Examine the prototype for the `PointVector` overloaded `[]` operator, and the use of the operator in `main`. There is a problem with the declaration – what is it? Write the corrected declaration.

(b) **(1 mark)**. Examine the declaration of the `PointVector` overloaded `<<` operator, and the use of this operator in `main`. There is a problem with the declaration – what is it? Write the corrected declaration.

(c) **(3 marks)**. Write the implementation of `PointVector::operator []`

(d) **(3 marks)**. Write the implementation of `PointVector::operator <<` such that the last line of `main` outputs: `(0,0) (3,2) (0,0)`

(e) **(1 mark)**. Is it necessary to write a destructor for class `Point`? Is it necessary to write a destructor for class `PointVector`? Write the prototype for any destructor(s) needed.

Question 5. (6 marks). *Linked Lists.*

Consider the following definitions of the two classes `listNode` (on left) and `linkedList` (on right) that are used to implement linked lists.

```
class listNode {
   private:
     int key;
     listNode* next;
   public:
     listNode();
     listNode(int k);

     int getKey();
     listNode* getNext();

     void setKey(int k);
     void setNext(listNode* n);

 };
```

```
class linkedList {
   private:
     listNode* head;

   public:
     linkedList();
     linkedList(const linkedList& src);
     ~linkedList();
         :
         :
     bool operator==(const linkedList& rhs);

};
```

Write the overloaded `operator==` of the class `linkedList` so that it returns true if the linked lists of its operands are identical, otherwise returns false. That is, given two variables of type `linkedList`, `firstList` and `secondList`, then (`firstList == secondList`) returns true if the two lists have the same number of nodes, and the nodes in `firstList` have exactly the same values of key as the nodes in `secondList` and in exactly the same order. Assume two empty lists to be equal by definition.

```
bool linkedList::operator==(const linkedList& rhs) {









}
```

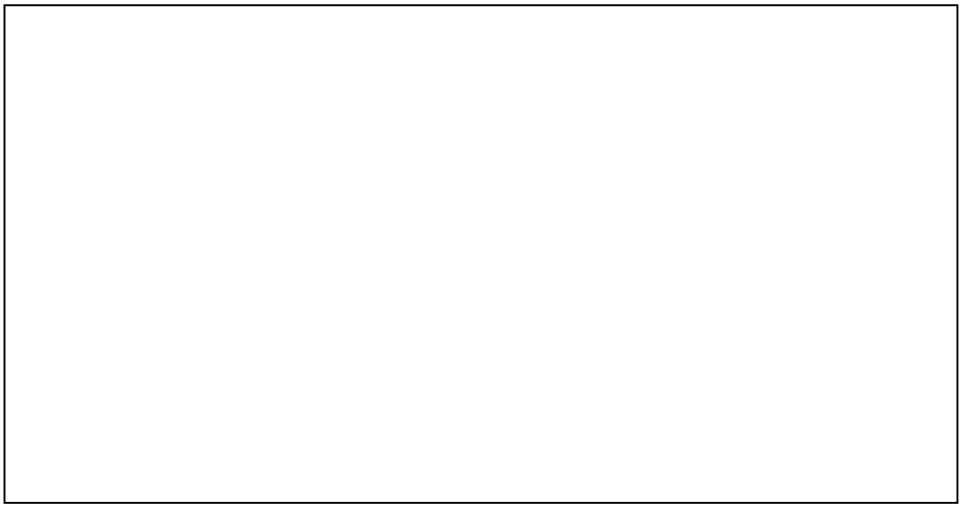
Question 6. (5 marks). *Recursion.*

Consider the following modified function that calculates Fibonacci numbers recursively:

```
int fib( int n, int depth ) {
    cout << setw( depth*3 ) << right << " "
         << "fib(" << n << "," << depth << ")"
         << endl ;
    if( n==0 ) return 0 ;
    if( n==1 ) return 1 ;
    int r1 = fib( n-1, depth+1 ) ;
    int r2 = fib( n-2, depth+1 ) ;
    return r1 + r2 ;
}
```

where "`setw(6)`" formats the next (and only the next) output to 6 characters and "`right`" right-adjusts the next output.

What is the output that is generated when the function `fib(3,0)` is called?



Question 7. (5 marks). *Tree Traversals.*

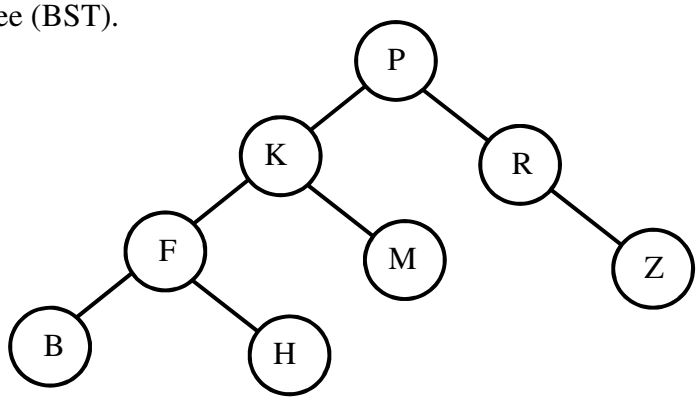
A tree ***T*** has the following inorder and preorder traversals:

```
Inorder  traversal:     1   8 12 14 16 20 26 28 30 40
Preorder traversal:     26 12  8  1 16 14 20 30 28 40
```

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

Question 8. (9 marks). *Binary Search Trees.*

The tree T shown here is a Binary Search Tree (BST).



(a) (3 marks). Insert a node with the key "C" onto the tree T and show the resulting tree.

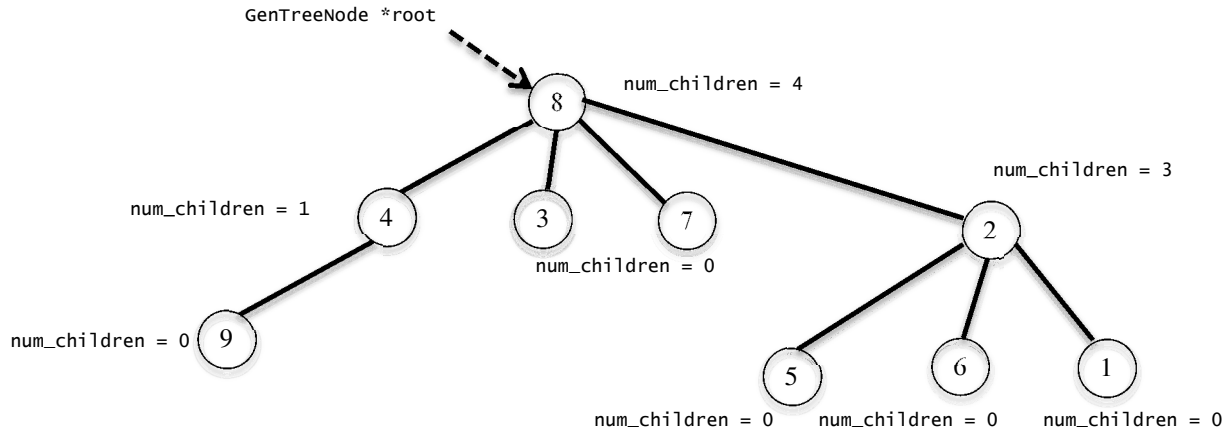
(b) (3 marks). Starting with the tree in part (a) after the insertion of "C", delete the node with the key "F" and show the resulting tree.

(c) (3 marks). Starting with the tree in part (b) after the removal of "F", re-insert the node with they key "F" onto the tree and show the resulting tree.

Question 9. (5 marks). *Trees.*

Consider the general tree structure illustrated below, and the corresponding class definition. This tree is a general tree where a node can have any number of children (0, 1, 2, or any integer, as stored in the member variable `num_children`), not just 2 as in a binary tree.

```
class GenTreeNode {
   public:
      // Key of the node
      int key;
      // The number of children
      int num_children;
      // Array [0..num_children-1] with pointers to children.
      // No NULL entries in the array.
      GenTreeNode *children;
      :
};
```



Write a recursive (non-member) function `preorder` to perform the preorder traversal of a general tree. The function is defined as: `void preorder (GenTreeNode* r);`

The function should print the key of each node and then traverse the child sub-trees staring with the leftmost (i.e. children[0]) child. For example, the preorder traversal of the above tree is: 8 4 9 3 7 2 5 6 1

```
void preorder (GenTreeNode* r) {









}
// This is how preorder is called
preorder(root);
```

Question 10. (3 marks). *Inheritance. [TSA]*

Consider the following definitions of the classes `Generic` and `Specific`.

```
class Generic {
   private:
      int _id;
   pubic:
      Generic(int id);
      ~Generic();
};


Class Specific : public Generic {
   private:
       float _data;
   public:
       Specific(float d);
       ~Specific();
};
```

How many data members do objects of type `Specific` have? [ ]

Question 11. (7 marks). *Inheritance.*

Consider the following definition and implementation for the `Shape` and the `Rectangle` classes.

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

class Shape {
    private:
        int _id;
        char* _name;
    public:
        Shape(int id, char* name);
        virtual ~Shape();
        int get_id();
        char* get_name();
        virtual void echo();
};


Shape::Shape(int id, char* name) {
    _id = id;
    _name = new char [strlen(name) + 1];
    strcpy(_name, name);
    cout << "Shape: constructor: " << _id << " " << _name << endl;
}


Shape::~Shape() {
    cout << "Shape: destructor: " << _id << " " << _name << endl;
    _name = NULL;
}


int Shape::get_id() {
    return _id;
}

char* Shape::get_name() {
    return (_name);
}


void Shape::echo() {
    cout << "Shape: echo: " << _id << " " << _name << endl;
}
```

```
class Rectangle : public Shape {
    private:
        float _width;
        float _length;
    public:
        Rectangle(int id, float width, float length, char *name);
        virtual ~Rectangle();
        int get_id();
        virtual void echo();
};

Rectangle::Rectangle(int id, float width, float length, char* name) :
Shape(id, name) {
    _width = width;
    _length = length;
    cout << “Rectangle: constructor: “ << get_id() << “ “
         << get_name() << endl;
}

Rectangle::~Rectangle() {
    cout << “Rectangle: destructor: “ << get_id() << “ “
         << get_name() << endl;
}

int Rectangle::get_id() {
    return Shape::get_id();
}

void Rectangle::echo() {
    cout << "Rectangle: echo: “ << get_id() << “ “
         << get_name() << endl;
    Shape::echo();
}

// main program
int main() {
    Shape      shap1(0,”Unknown”);                  /* 1  */
    Shape*     shape_ptr = &shap1;                  /* 2  */
    Rectangle  rect1(10, 3.0, 6.3, "small");        /* 3  */
    Rectangle* rect_ptr = &rect1;                   /* 4  */
                                                    /* 5  */
    cout << shape_ptr->get_id() << endl;            /* 6  */
    shape_ptr->echo();                              /* 7  */
    rect_ptr->echo();                               /* 8  */
    *shape_ptr = *rect_ptr;                         /* 9  */
    rect_ptr->echo();                               /* 10 */
    shap1 = rect1;                                  /* 11 */
    cout << shape_ptr->get_id() << endl;            /* 12 */
    rect_ptr->echo();                               /* 13 */
}                                                   /* 14 */
```

Show the output produced by each line of the program in the table below. If a line has no output, write `N/A` (i.e., do NOT leave blank).

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

Question 12. (4 marks). *Inheritance.*

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:
    Boodle *bp ;
  public:
    A() ;
    virtual ~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() ;
    virtual ~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

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




}
```

Question 13. (4 marks). *Template Functions.*

The following function searches for a value in an integer array and returns true if found along with the index of the location of the found value through the index parameter. This function works correctly for integers. Re-write the function to make it into a template function so that search may also be used for doubles, characters, unsigned integers, etc.

```
bool search( int *a, int from, int to,
                int value, int& index)
{
    if( from > to ) return false ;
    int mid = (from + to) / 2 ;
    if( value == a[mid] ) {
        index = mid ;
        return true ;
    }
    if( value < a[mid] )
        return search( a, from, mid-1, value, index ) ;
    else
        return search( a, mid+1, to, value, index ) ;
}
```

Write your answer here.

Question 14. (6 marks). *Template Classes.*

Consider the following definitions of the class `Strange`.

```
template<class T>
class Strange {
   private:
      T a;
      T b;
   pubic:
      Strange();
     ~Strange();
      :
};
```

(a) **(1 mark)**. Write a statement that declares `myStrangeObject` as an object of type `Strange` such that the private data members `a` and `b` are of the type `int`.

(b) **(1 mark)**. Write the function prototype of the overloaded `operator==` function as a member function of the class `Strange`.

(c) **(2 marks)**. Assuming that two objects of type `Strange` are equal if their corresponding members `a` and `b` are equal, write the implementation of the function `operator==` for the class `Strange`.

(d) **(2 marks)**. Will your implementation of `operator==` in part (c) above work for any type of the private members `a` and `b`? Justify your answer with a one-sentence explanation.

Question 15. (6 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) (3 marks). The size of the input is `n`.

```
int a = 0;
for (int i = 0; i < n; ++i) {
  for (int j = 0; j < i*i; ++j) {
    for (int k = 0; k < 1000000; ++k) {
        a = a + 1;
  }
}
```

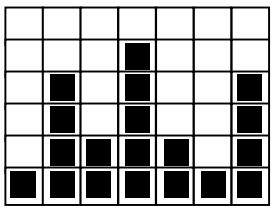
(b) (3 marks). The size of the input is `n`. Assume for simplicity that `n` is a power of four. Write the recurrence equation and then solve it.

```
int recursive(int n) {
   int x,y,w,z;

   if (n <= 1) return 0;
   else {
       x = recursive (n/4);
       y = recursive (n/4);
       w = recursive (n/4);
       z = recursive (n/4);
       return (x+y+w+z);
   }
}
```

Question 16. (5 marks). *Efficient Algorithms*

The image below shows an `n` by `n` grid that is stored in memory as a two dimensional array. The element at row `i` column `j` can be indexed in constant time using `grid[i][j]`. Each cell in the grid contains either a `1` or a `0` (indicated in the figure with black and white blocks). In any column, all the one's appear below any of the zeros. Given such a grid, design an O(n) algorithm for finding any column with the most ones (tallest black tower). Note there are $O(n^2)$ cells, so you cannot check every cell in the grid.



Then implement your algorithm as a function `tallest()` that returns the column with the most ones:

```
int tallest( int **a, int n /* matrix size */) {









}
```

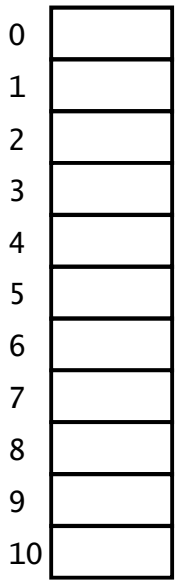
Question 17. (3 marks). *Hash Tables*.

Assume a hash table with size T = 11 with the following hash function:

h: index = (key mod T) (note: mod is the modulus operator, % in C++)

Show the contents of the hash table after the following operations have been performed. Indicate next to each operation the number of ***probes*** performed. Assume that linear probing is used to resolve collisions and that the hash table is initially empty.

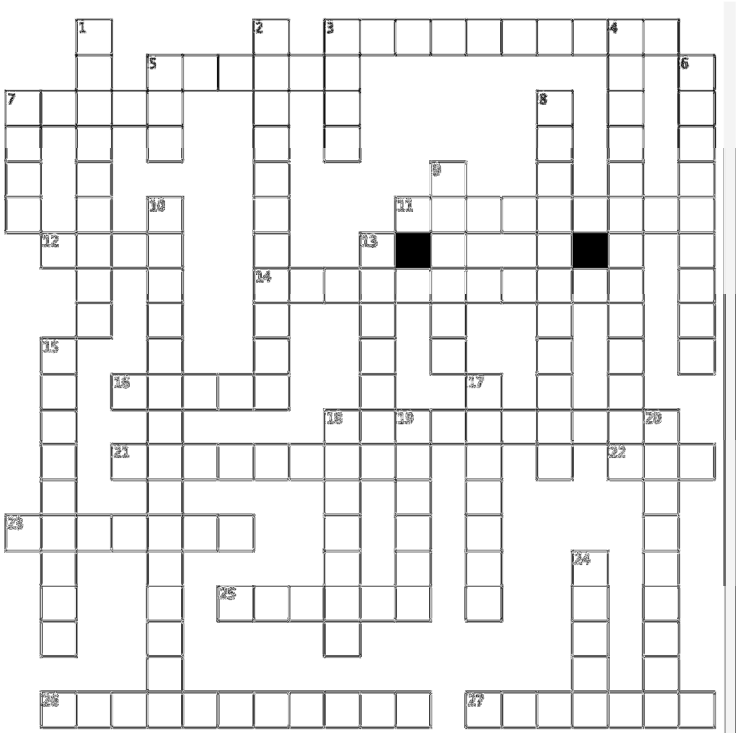
1. insert 14
2. insert 26
3. insert 4
4. delete 26
5. search 4
6. insert 37



Question 18. (2 marks). *Crossword Puzzle. [TSA]*

This question is dedicated to the student who sat at the back of the class solving crossword puzzles. Have a blast!

Solve the crossword puzzle below.



Across

3. they are really efficient for search
5. you use it to free memory
7. unix command for setting permissions
11. visiting each node once
12. ensures executable is up to date
14. a really bad complexity class
16. they have roots and leafs
18. function called when object is deleted
21. a really good sorting algorithm
22. you use it to dynamically allocate data
23. instances of classes
25. members accessible by all functions
26. function called when object is created
27. combining object files to make executable

Down

1. using ___ std;
2. you avoid these drippings at any cost
3. memory area for dynamic data
4. hiding implementation
5. the debugger used in lab assignment 1
6. C++ mechanism for code reuse
7. __ << x << endl;
8. C++ mechanism for code reuse
9. nonmember function that can access private members
10. cannot be invoked without an object
13. variable that holds address of data
15. program used to check your lab submissions
17. members not accessible by non-members
18. this loop iterates at least once
19. variables that persist beyond their scope
20. used to implement divide-and-conquer
24. automatic variables reside there

THIS PAGE IS INTENTIONALLY BLANK FOR ANSWER OVERFLOWS