University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2010

Final Examination

Examiner: T.S. Abdelrahman, M. Gentili, and M. Stumm

Duration: Two and a Half Hours

This exam is CLOSED books 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. __________

Q2. __________

Q3. __________

Q4. __________

Q5. __________

Q6. __________

Q7. __________

Q8. _________

Q9. _________

Q10. ________

Q11. _________

Q12. _________

Q13. _________

Q14. _________

Q15. _________

Q16. _________

Total □

Question 1. (12 marks). *General.*

Answer the following questions by circling either **Yes** or **No**, or by providing a **very brief** and **direct** answer when indicated.

(a) **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.

(b) **Yes or No**? A pure (abstract) virtual function of a base class may or may not be implemented in the base class.

(c) **Yes** or **No**. An `O(log(n))` algorithm is always faster than an `O(n)` algorithm when `n>1`?

(d) **Yes** or **No**. If a `C++` class requires a destructor to de-allocate some resource, then most likely it needs a copy constructor and `operator=` definition as well.

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

(f) **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;").

(g) **Yes** or **No**? The largest value of a binary search tree is always stored at the root of the tree.

(h) **Yes** or **No**? In a binary tree, every node has exactly two children.

(i) **Yes** or **No**? To delete a dynamically allocated binary search tree, the best traversal method is *postorder*.

(j) **Yes** or **No**? It is possible to implement inserting an element onto an unsorted list in O(1) time.

(k) How would you force a derived class to implement a virtual method from the base class?

(l) What Unix command would you type to create a new directory called `ece244`?

Question 2. (4 marks). *Pointers.*

Provide two possible ways in which a function `update_ptr` may change the value of a pointer of type `int*`. The pointer is defined outside the function and is passed to it as an argument. Assume the formal argument of `update_ptr` is a variable called `p` and that the function is called with an actual argument called `ptr`. Thus, the function changes the value of `ptr`. For each way, show the function prototype and how the function should be invoked.

Write your answer below.

First way:

function prototype: `void update_ptr(                    );`

invocation (i.e., function use): `update_ptr(                    );`

Second way:

function prototype: `void update_ptr(                    );`

invocation (function use): `update_ptr(                    );`

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

Consider the C++ code shown below.

```
#include <iostream>
int main() {
    int* p1;
    p1 = new int;
    *p1 = 5;
    int& q = *p1;
    q = 8;
    cout << *p1 << endl;    // cout # 1
    *p1 = 14;
    cout << q << endl;      // cout # 2
    int* p2 = &q;
    *p1 = 4;
    cout << q << endl;      // cout # 3
    q = 16;
    cout << *p2 << endl;    // cout # 4
    return (0);
}
```

Show what is printed by each of the "cout" statements in the main function. To simplify providing an answer, each "cout" statement is given a number in the comments in the above code, so you can provide your answer in the table below.

cout #	Output
1	
2	
3	
4	

Question 4. (6 marks). *Constructors/Member Methods.*

Consider the following class definition. The numbers listed on the left are not part of the code; they are there for reference.

```
 1: class Golfer {
 2:    private:
 3:       char * fullname;
 4:       int games;
 5:       int * scores;
 6:   public:
 7:       Golfer ();
 8:       Golfer (const char * name);
 9:       Golfer (const char * name, int g);
10:       Golfer (const Golfer & g);
11:       ~Golfer ();
12: };
```

What class methods (if any) would be invoked by each of the following statements?

Statement	Class Method (write line number)
`Golfer nancy;`	
`Golfer lulu (“little lulu”);`	
`Golfer roy (“Roy Hobbs”, 12);`	
`Golfer *par = new Golfer ();`	
`Golfer next = lulu;`	
`*par = lulu;`	

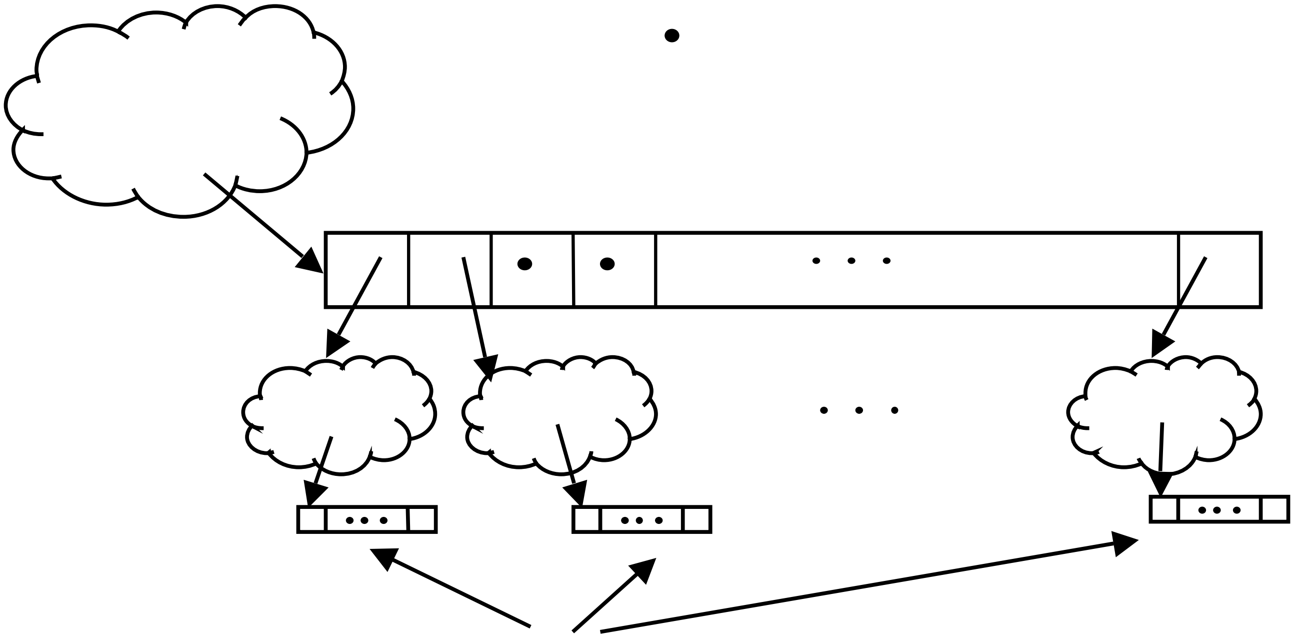
Question 5. (8 marks). *Memory Management*.

Consider the following definition of the two classes, `database` and `element`.

```
class database {
    private:
         int n; // The size of the array ‘thearray’
         element** thearray;
    public:
         :
         :
         :
};

class element {
    private:
         int count;
         char* name;
    public:
         :
         :
         :
};
```

The public functions of each of the two classes include the constructors, accessor methods and the destructor.

A `main` function uses these class definitions to construct a `database` object called `mydatabase` and many `element` objects as shown in the figure below.


means NULL
n
thearray
0
1
2
3
dynamically allocated array
n-1
mydatabase
dynamically allocated objects of type element
count
name
count
name
count
name
dynamically allocated char arrays

Write the destructors of the two classes, `database` and `element` such that no memory leaks exist when the object `mydatabase` goes out of scope. Note that all variables are dynamically allocated as indicated in the above figure. However, `mydatabase` is an automatic variable. Write your code in the space provided below.

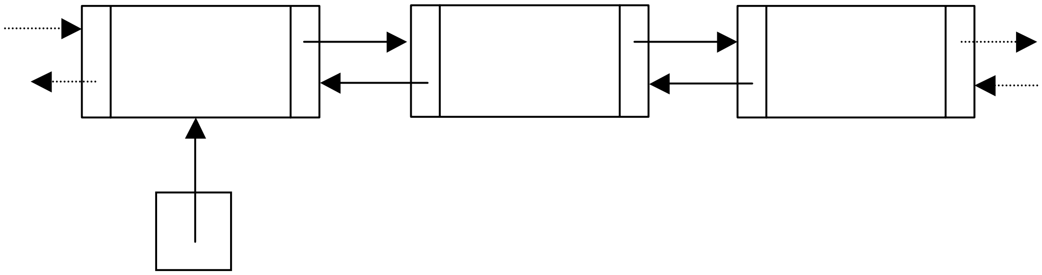
```
database::~database() {







}
```

```
element::~element() {







}
```

nodePointer
value=15 value=18 value=23

Question 6. (6 marks). *Pointers and Linked Lists***.**

A doubly-linked list is a linked list in which each node has a pointer to its successor and a pointer to its predecessor. Assume that a doubly-linked list has been constructed for you, as shown below. You have been passed a pointer, `nodePointer`, to an element which resides somewhere in the middle of a very large list:



Each of the Nodes has the following structure:

```
struct Node {
     int       value;     // The value contained in the Node
     Node *    next;      // Points to the next Node in the list
     Node *    prev;      // Points to the previous Node
};

Node *     nodePointer;
```

Your task is to write the lines of code which would free the space of the node **with value=18**, using the `delete` operator, after properly removing it from the list. The remaining elements in the list must be properly linked to one another after the deletion.

Note that you are **not** being asked to write a complete function: just the lines that will delete the appropriate element (i.e., the Node with value=18) from the list that is shown above, starting only with `nodePointer`. You may **not** allocate or use any additional variables (including pointers).

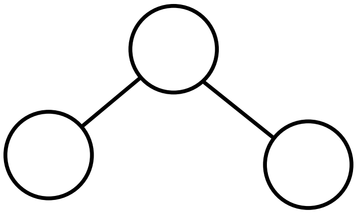
Hint: You can change the value of `nodePointer`. Four lines are all it takes.

B

A C

Question 7. (6 marks). *Tree Traversals*.

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)! Excessively 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);
```

```
int rlen (char* s) {
```

Question 8. (6 marks). *Recursion.*

Write a recursive function that finds the length of a c-string. That is, given a properly NULL-terminated c-string `s`, the function returns the number of characters in the c-string, excluding the NULL character (i.e., the '\0' character).

```
}
```

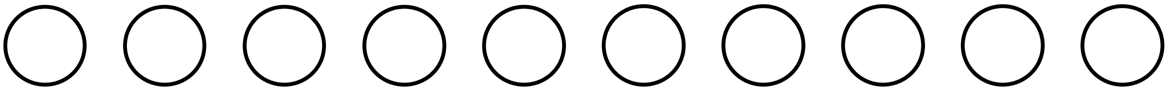
Your implementation of the function must be recursive.

Hint: This is one time when you want to use pointer arithmetic. Recall that if `p` is a pointer to the first element of a character array, then `p+1` is a pointer to the next element. Think of the simplest basis!

10 14 8 4 2 12 25 6 28 13

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

Consider the following sequence of binary tree nodes. The key of each node appears inside the node.



(a) **(2 marks)**. Draw the binary search tree that results from the successive insertion of the nodes in the order in which they appear above from left to right.

(b) **(2 marks)**. Delete the node with the key 14 in the tree in part **(a)** above and redraw one of the possible resulting binary search trees.

(c) (2 marks). Re-insert the node with the key 14 in the tree in part **(b)** above and re-draw the resulting binary search tree.

(d) (2 marks). Give the inorder traversal of the tree in part (c).

Question 10. (4 marks). *Inheritance.*

Consider the following code.

```
class A {
   public:
      virtual void iAm();
      void callerSees();
 };

void A::iAm() {
   cout << "class A" << endl;
}

void A::callerSees() {
   cout << "class A" << endl;
}


class B : public A {
    public:
      void iAm();
      void callerSees();
};


void B::iAm(){
   cout << "class B" << endl;
}

void B::callerSees(){
   cout << "class B" << endl;
}

int main() {
   B b;
   A* a = &b;
   a->iAm();
   a->callerSees();
}
```

What is the output of `main()`? Write your answers in the table below, in the order in which it is printed by the program, one output line per row. Note that the table may have more rows than the number of lines printed by `main()`.

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

Consider the following code.

```
class Base {
   public:
      Base();
      ~Base();
};

Base::Base() {
   cout << "Base::Base()" << endl;
}

Base::~Base() {
   cout << "Base::~Base()" << endl;
}



class Inherited : public Base {
   private:
      char* note;

   public:
      Inherited(char* newNote);
      ~Inherited();
};

Inherited::Inherited(char* newNote) {
   cout << "Inherited::Inherited(" << newNote << ")" << endl;
   note = new char[strlen(newNote) + 1];
   strcpy (note, newNote);
}

Inherited::~Inherited() {
   cout << "Inherited::~Inherited(" << note << ")" << endl;
   delete note;
}



int main() {
   Inherited a("stack");
   Base * bp = new Inherited("heap");
   delete bp;
}
```

(a) What is the output of `main()`? Write the output in the table below, in the order in which it is printed by the program, one output line per row. Note that the table may have more rows than the number of lines printed by `main()`.

(b) The program contains a memory allocation error. Describe in one sentence what the error is.

(c) How would you fix the error?

Question 12. (7 marks). *Inheritance.*

Consider the C++ code shown below for a base class xValue.

```
class xValue {
    private:
        int x;
    public:
        bool valid;
        xValue(int xv);
        virtual ~xValue();
        void double2();
        virtual void negate();
        virtual void print()=0;
};

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

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

void xValue::double2() {
    x = 2*x;
}

void xValue::negate() {
    x = -1*x;
}

void xValue::print(){
    cout << “x value = “ << x << “ “;
}
```

Now consider the following class, xyValue, which derives from xValue.

```
class xyValue : public xValue {
    private:
        int y;
    public:
        xyValue(int xv, int yv);
        virtual ~xyValue();
        void double2();
        virtual void negate();
        virtual void print();
};


xyValue::xyValue(int xv, int yv):xValue(xv) {
    y = yv;
}
```

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

void xyValue::double2() {
    xValue::double2();
    y = 2*y;
}

void xyValue::negate() {
    xValue::negate();
    y = -1*y;
}

void xyValue::print(){
    xValue::print();
    cout << “, y value = “ << y << endl;
}
```

Now answer the following questions, based on the above definitions and implementations of the two classes. You should assume each part of the questions to be independent of the others.

(a) (3 marks). Indicate by placing an X in the appropriate column whether each of the following code segments is *correct* code (i.e., compiles correctly) or *incorrect* code (i.e., produces a compile error).

	Correct	Incorrect
xValue v0;		
xValue v1(2);		
xyValue v2(3,5);		

void xyValue::negate() {
    x = -1*x;
    y = -1*y;
}		

xyValue v3(10,10);
v3.valid = false;		

xyValue v4(5,7);
v4.x = 4;		

(b) (2 marks). Indicate the output produced by the following code segment.

	Output
`xyValue* p2=new xyValue(3,8);`	
`xValue*  p1;`	
`p1 = p2;`	
`p2->double2();`	
`p2->print();` →	
`p1->double2();`	
`p1->print();` →	
`p2->negate();`	
`p2->print();` →	
`p1->negate();`	
`p1->print();` →	

(c) (2 marks). Indicate the constructors and the destructors invoked by each of the following code segments, *in the order in which they are invoked*. Assume each segment is independent of the other segments. Indicate a constructor or a destructor by its name; e.g., `xyValue` or `~xyValue`.

	Constructors/Destructors

`if (..) { //condition is true`
`   xyValue a1(10,10);`
`}`	

`xValue* p = new xyValue(1,2);`
`delete p;`	

Question 13. (8 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) (4 marks) The size of the input is `n`.

```
w=0;
for (int i=0; i < n; ++i) {
  for (int j=0; j < 10; ++j) {
    for (int k=0; k < n*n; ++k) {
       w = w + 1;
    }
  }
}
```

T(n) = □

(b) (4 marks). Assume for simplicity that `n` is a power of two.

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

   if (n <= 1) return 0;
   else {
       x = recursive (n/2);
       y = recursive (n/2);
       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) =` □

Question 14. (6 marks). *Complexity Analysis.*

Suppose that you have an array A of integers and the integers are not in any sorted order. However, the array contains duplicate items and you want to create another array B that contains all the items in A but without any duplicate items. The integers in B are not to be in any sorted order. Three algorithms are described below. What is the average case running time in big-O notation of each algorithm? State any assumptions you make.

(a) **(2 marks)**. For each item in array A, copy the item into another array B unless the item already exists in B.

(b) **(2 marks)**. Sort A using Quicksort. Now repeat part **(a)**.

(c) **(2 marks)**. Insert each element of A in a hash table unless a duplicate item already exists in the hash table. At the end, copy all items from the hash table into array B. Assume a low density (i.e., load factor) for the hash table.

Question 15. (6 marks). *Hash Tables.*

Perform the following inserts into the 10-element hash table below, using closed hashing with linear probing and the following hash function, h(k):

h(k) = k % 10

Show the contents of the array after all inserts have completed.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

insert 17
insert 408
insert 543
insert 88
insert 989
insert 500

Question 16. (3 marks). *The Make Utility.*

Consider the following Makefile:

```
1
2   all: maintool findtool mixtool
3
4   tree.o: tree.cpp list.h
5   g++ -g -Wall -c tree.cpp -o tree.o
6
7   list.o: list.cpp list.h
8   g++ -g -Wall -c list.cpp -o list.o
9
10  findtool.o: findtool.cpp findtool.h
11  g++ -g -Wall -c findtool.cpp -o findtool.o
12
13  findtool: findtool.o
14  g++ findtool.o -o findtool
15
16  mixtool: tree.o list.o list.h
17  g++ list.o tree.o -o mixtool
18
19  maintool: findtool.o tree.o list.o
    g++ tree.o findtool.o -o maintool
```

The following table shows several invocations of the Make utility using the above correct Makefile. For each invocation, indicate the commands that are executed as a result of the invocation, *in the order in which they are invoked*. If no commands are executed, write "None". In your answer, only provide the line numbers corresponding to the commands that are executed.

Assume that no files generated as a result of calling make exist before the first call of make. Also, the invocations of Make are performed *in the order shown in the table (i.e., the different commands are not independent).*

Assume that the Makefile exists in the same directory as all the .cpp and .h files.

Make Invocation	Commands Executed (indicate line number)
make maintool	
make mixtool	
make findtool	