University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2008

Final Examination

Examiner: C. Gibson, A. Goel, and M. Stumm

Duration: Two and a Half Hours

No aids allowed. No books. No notes. No calculators. No computers. No communicating devices.

Do not remove any sheets from this test book. Answer all questions in the space provided. No additional sheets are permitted.

There are 15 questions. The weight of each question is the same. Work independently.

Write your name and student number in the space below. Do the same on the top of each sheet of this exam book.

Last Name: ___________________ First Name: _________________________

Student Number: ______________________

Q1. __________	Q9. __________
Q2. __________	Q10. _________
Q3. __________	Q11. _________
Q4. __________	Q12. _________
Q5. __________	Q13. _________
Q6. __________	Q14. _________
Q7. __________	Q15. _________
Q8. __________	

Total $\boxed{}$

Name: ______________________________ Student Number: __________________

1. Warmup

a. Consider the following code:

```
class Data {
  private:
     int a;
  public:
     int get() { return a; }
};

int main() {
     Data data;
     return 0;
}
```

Modify the code above so that `data` is declared as a global variable.

b. Identify what is wrong with the following code and fix the error(s):

```
class C {
public:
  int x;
};

void main(void) {
   int x = 78;
   C c;

   c->x = x+22;

   cout << c->x << endl ;
}
```

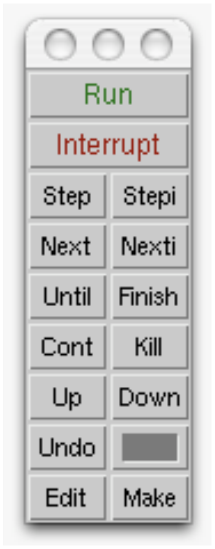
c. Identify what is wrong with the following code and fix the error(s):

```
const int MAX 10

int list[MAX];

void main(void) {
   int sum = 0;

   for (int i = 0; i < MAX+1; i++) {
      list[i] = i;
   }
   for (int i = 0; i < MAX+1; i++) {
      sum += list[i];
   }
   cout << sum << endl ;
}
```

Name: ________________________________ Student Number: ____________________

2. GDB and Unix

a. ***Briefly*** describe (one sentence) what the “Step” command does in either gdb or DDD when debugging a running program:



Answer:

b. Assume your current working directory is your home directory. What Unix command(s) would you need to execute to create a new directory “Lab4” under the existing “ece244” directory.

c. What command do you need to execute in order to make the contents of this directory not readable by anyone else but yourself?

d. You are testing a program that reads from the standard input until end of file is reached. When you have completed providing input, do you enter Control-C or Control-D?

e. What command do you issue in order to remove a file?

3. Scope and parameters

Show the output of the following program in the box provided below. Show your work for partial credit.

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

char confuse (char c1);
void mixup (char &c1, char c2, char c3);

char c4 = 'F';

int main () {
   char c1 = 'H', c2 = 'C', c3 = 'E';
   mixup( c2, c1, c3);
   {
      char c3 = 'G';
      cout << c1 << c2 << c3 << c4 << endl;
   }
   cout << confuse(c4) << c1 << c2 << c3 << c4 << endl;
   return 0;
}

char confuse (char c1) {
   const char c2 = 'O';
   char c3 = c1;
   if ( c1 >= 'N' ) return c3;
   else             return c2;
}

void mixup (char &c1, char c2, char c3) {
   cout << c1 << c2 << c3 << c4 << endl;
   c1 = confuse(c3);
   c2 = 'S';
   c3 = 'I';
   c4 = confuse(c2);
}
```

Output line 1:	
Output line 2:	
Output line 3:	
Output line 4:	

Name: ______________________________ Student Number: __________________

4. 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. If you make any assumptions, state them.

a. The size of the input is `n`.

```
int
whatisit(int x, int n)
{
      int m;
      if (n <= 0)
            return 1;
      m = whatisit(x, n / 2);
      if (n % 2)
            return x * m * m;
      else
            return m * m;

}
```

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) =` $\boxed{}$

Name: ______________________________ Student Number: __________________

b. The size of the input is n.

```
int
whatisit(int n)
{
    int x;

    if (n <= 1) {
        return 1;
    }
    if (n % 2) {
        x = whatisit(n/2) + whatisit(n/2 + 1);
    } else {
        x = 2 * whatisit(n/2);
    }
    return x;
}
```

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) = $\boxed{}$

Name: ______________________________ Student Number: __________________

5. Programming

Write the implementation of the following function to output to the standard output the null-terminated string, passed in as a character array parameter, so that the letters of the string (prior to the null-terminating character) are output backwards where the last character is output first, the second to last character is output next, and the first character is output last and do so **without using any types of loops**, such as for loops or while loops, **and without using strlen() or any other str… functions**. For example, if the array passed in as a parameter contains the characters 'H', "e', 'l', 'l', 'o', Ø, then the function should output the string the characters 'o', 'l', 'l', 'e', and 'H'.

```
void output_backwards( const char * cp ) {
      // write your code here…

}
```

6. QuickSort

The following is a not-so-nice, but correctly functioning implementation of a select-and-shuffle algorithm called by quicksort. (This function is sometimes also called "partition".)

```
int selectAndShuffle(int * a, int left, int right)
{
   int i=left-1, j=right;
   int pivot=a[right];
   for (;;) {
      while (a[++i] < pivot);
      while (pivot<a[--j])
         if (j==left) break;
      if (i>=j) break;
      swap(a[i], a[j]);
   }
   swap(a[i], a[right]);
   return i;
}
```

In this implementation, the pivot value is selected as the rightmost array member. Assume the function `swap()` swaps the values of the two arguments correctly.

a. Given an array *a[6]=[5,5,5,5, 5,5]* , how many `swap()` function calls will be executed during the partition? The input parameters for the partition call are: *left=0*, and *right=5*.

Answer:

b. if we change line 6 to *while(a[++i]<=pivot);*, will the algorithm still work correctly?

Answer:

7. Classes with Pointers

The following code shows partial declarations of a node class and a binary tree class.

```
class Node {
  public:
      Node(int);
      Node(const Node &);
      int value;
      Node *left;
      Node *right;
};

class Tree {
  public:
      Tree();
      ~Tree();
      Node *root;
      Tree(const Tree &);
};
```

Assume that the code above compiles correctly and that all member functions of the Node and Tree class are defined, except the copy constructor of the binary tree class. The copy constructor must do a deep copy of the binary tree. The following code implements the binary tree copy constructor using a helper member function called TreeCopy. Your job is to write the TreeCopy member function. Make sure to declare the types of all parameters in TreeCopy.

```
Tree::Tree(const Tree &t) {
      TreeCopy(&root, t.root);
}
```

Name: ______________________________ Student Number: __________________

8. Exception Handling

Assume that you are given the following definitions and the main program.

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

class BadPointer {
   public:
      void print();
};

class OutOfBound {
   public:
      void print();
};

void BadPointer::print()
 { cout << "BadPointer" << endl; }

void OutOfBound::print()
 { cout << "OutOfBound" << endl; }

void f2(int *ip, int index,
                 int size)
{
   if (index < 0 || index >= size)
      throw OutOfBound();
   cout << "value = " << index
        << endl;
}
```

```
void f1(int *ip, int size) {
   try {
      if (ip == NULL)
         throw BadPointer();
      f2(ip, 10, size);
   } catch (OutOfBound e) {
      cout << "f1: "; e.print();
   }
   if (size > 0) f2(ip, size, 5);
}

int main() {
   int i;
   int *ip = NULL;

   cin >> i;
   if (i > 0) ip = new int[i];
   try {
      f1(ip, i);
   } catch (OutOfBound e) {
      cout << "main: "; e.print();
   } catch (BadPointer e) {
      cout << "main: "; e.print();
   }
   cout << "end" << endl;
   return 0;
}
```

What is printed if the user enters the following input:

Input	Output	Input	Output
20		1	
10		0	
5		-1	

Name: ______________________________ Student Number: __________________

9. Inheritance

Consider the C++ code shown below for three classes, node, lnode and cnode, and the main program shown on the next page.

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

// Calling virtual functions on a pointer, reference, not copy
// Three level hierarchy

class node {
  public:
    node(int n) { number = n; }
    int get_num() { return number; }
    virtual ~node() {};
  private:
    int number;
};

class lnode : public node {
  public:
    lnode(int l): node(10) { length = l; }
    virtual int get_num()  { return 2 * node::get_num(); }
    virtual int get_len()  { return length; }
    virtual ~lnode()       { cout << "~lnode" << endl; }
  private:
    int length;
};

class cnode : public lnode {
  public:
    cnode(int l, char *c);
    int get_len() { return 2 * lnode::get_len(); }
    virtual ~cnode();
  private:
    char *color;
};

cnode::cnode(int l, char *c) : lnode(l)
{
    color = new char[strlen(c) + 1];
    strcpy(color, c);
}

cnode::~cnode() {
    delete color;
    cout << "~cnode" << endl;
}
```

Name: ______________________________ Student Number: ___________________

```
int main()
{
    cnode c(12, "red");
    lnode &l = c;
    lnode *lp = &c;
    node &n = c;
    node *np = &c;

    cout << "l.get_num: " << l.get_num() << endl;
    cout << "lp->get_num: " << lp->get_num() << endl;
    cout << "n.get_num: " << n.get_num() << endl;
    cout << "np->get_num: " << np->get_num() << endl;

    cout << "l.get_len: " << l.get_len() << endl;
    cout << "c.get_len: " << c.get_len() << endl;
    cout << "lp->get_len: " << lp->get_len() << endl;

    lnode ln = c;
    cout << "ln.get_num: " << ln.get_num() << endl;
    cout << "ln.get_len: " << ln.get_len() << endl;

    return 0;
}
```

Show the output of the main program below.

Line	Output
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
15	

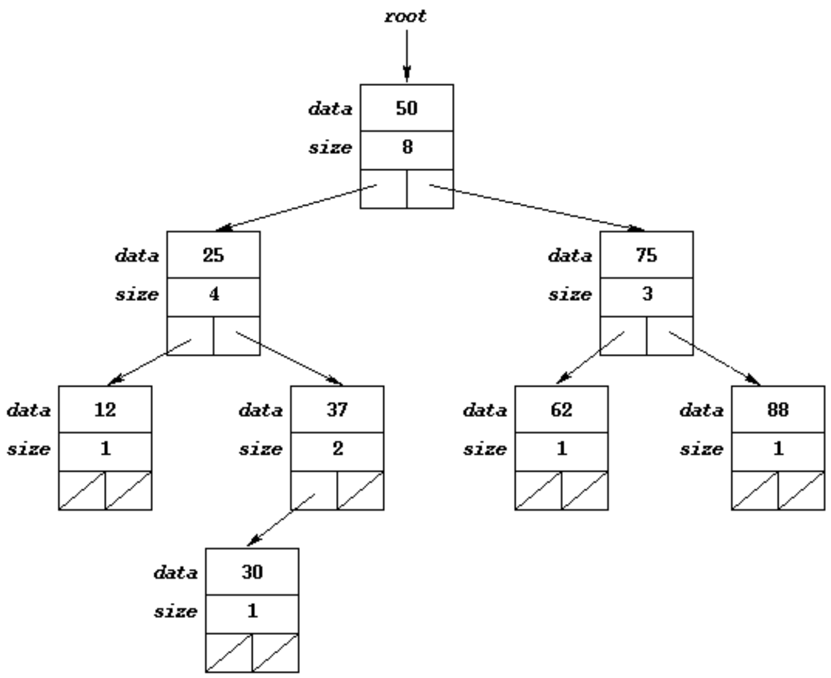
Name: ______________________________ Student Number: __________________

10. Binary Trees II

Assume that binary trees are implemented using the following declarations.

```
struct TreeNode
{
    int        info;     // data stored
    int        size;     // size of subtree (root included)
    TreeNode * left;     // left subtree
    TreeNode * right;    // right subtree

    TreeNode(int val, TreeNode * lptr, TreeNode * rptr)
          : info(val), size(1), left(lptr), right(rptr) { }
};
```

a. Write a function `SetSize()` in the space provided on the next page where its header is given. `SetSize()` should set the `size` field of every node in its tree parameter to the number of nodes in that node's sub-tree (including itself). The number of nodes in a tree is equal to the number of nodes in its left sub-tree plus the number of nodes in its right sub-tree plus one. For example, the following picture shows the result of the call `SetSize(root)`:



b. Assume that tree *T* is a binary search tree ordered by the values in the nodes' `info` fields, and that there are no duplicate `info` values. The *k*th value in a binary search tree is the *k*th smallest value in the tree. For example, the tree shown above includes the data values 12, 25, 30, 37, 50, 62, 75, 88.

Write a function `FindKth()` in the space provided on the next page where its header is given. `FindKth()` is to return the *k*th value in a binary search tree. Assume that the `size` fields in all nodes of the tree have been correctly initialized. One way to determine the location of the *k*th value is as follows: Consider the size of the left subtree of the current node. If k is equal to the size of the left subtree + 1, the kth value is in the current node. If k is less than the size of the left subtree + 1, the kth value is in the left subtree. Otherwise, the kth value is in the right subtree.

```
void SetSize(TreeNode * t)
{

}

int FindKth(TreeNode * t, int k)
  // assume: t is not NULL,
  //         the size fields of all nodes in t are correct.
  //          1 <= k <= t->size
{

}
```

Name: ________________________ Student Number: ___________________

11. Make

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*. 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.

Recall that the `touch` command simply updates the timestamp of its argument to the current time, as if the file had been modified at the time `touch` was invoked.

Make Invocation	Commands Executed (indicate line number)
`make maintool`	

`touch list.h`
`make mixtool`	

`make findtool`	
`make all`	

Answer:

Answer:

Name: ______________________________ Student Number: __________________

12. Inheritance

Consider the following inheritance hierarchy:

```
class A{
  protected:
    int x, y;
  public:
    int z;
}
class B: public A{
  private:
    int a, b, c;
}
```

a. How many data members does B have?

b. How many of B's data members are directly *accessible* in B?

13. Lab 7

Recall that the `Record` class in Lab 7 had fields to represent the number (key) of an individual, and it had the following member functions associated with it:

- `Record ()`. This is the default constructor. It creates an empty record.
- `virtual ~Record()`. This is the destructor. It `delete`s all dynamic components of the record.
- `void setNumber(unsigned int Num)`. This function sets the number in the record to `Num`.
- `unsigned int getNumber()`. This function returns the number in the record.
- `virtual void print()=0`.

You then derived a class `studentRecord` from the class `Record` that added data members to store the marks obtained and the member functions `setMark()`, `getMark()`, `setFirstName()`, `setLastName()`, `getFirstName()`, `getLastName()`, and `print()`, all with appropriate arguments.

a. Briefly describe why the print() function in Record was declared to be virtual.

Name: ______________________________ Student Number: __________________

b. In the space below, provide the definition (as might be contained in a `.h` file) of a class `AuditorRecord` that is derived from `Record` for Auditors that do not receive marks.

14. Dynamic memory allocation
Consider the following program fragment that makes use of `Suzy` class:

```
Suzy *func1( Suzy a ) {
   Suzy b = a ;
   Suzy *c = new Suzy( b ) ;
   return c ;
}
void func2( Suzy & d ) {
   Suzy e ;
   Suzy *f = new Suzy() ;
   Suzy *g = &d ;
   Suzy *h = func1( *g ) ;
   // point X
   return ;
int main() {
   Suzy i ;
   Suzy j = new Suzy( i ) ;
   func2( i ) ;
   return 0 ;
}
```

How many objects exist in memory when the program reaches ***point X***?

0
1
2
3
4
5
6
7
8
9
10

Name: ______________________________ Student Number: __________________

15. Hash Tables

Assume you are given a hash table with size M = 11, and the following hash function:

h(key) = (key % M) (Note: the "%" is the modulus operator in C++)

Assume that **quadratic probing** is used to resolve key collisions, and that the hash table is initially empty. If *i* is the initial hash index generated by the hash function above, quadratic probing examines locations $i + (k)^2$, where $k = 0, 1, 2, 3, \ldots$ until an empty location is found.

Show the final contents of the hash table after all of the following operations have been performed, in the sequence shown. Next to each operation, indicate the number of probes performed when inserting that value.

insert 7

insert 3

insert 18

insert 29

insert 14

insert 40
