UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS105 — Computer Fundamentals
Final Examination — April, 2009

Examiner: John Carter

Duration: 2.5 h

Exam Type: A
This is a "closed book" examination; no aids are allowed.

Calculator Type: 4
No calculators are allowed.

All questions are to be answered on the examination paper. If the space provided for a question is insufficient, extra space is provided at the end of the examination. If you use this extra space, please indicate clearly which question(s) you have answered there.

The examination has 12 pages, including this one.

The marks allocated to the questions, out of a total of 135, are shown in the question headings.

You must use the C programming language to answer programming questions.

Name _____

Student Number _____ ecf login _____

MARKS

1	2	3	4	5	6	7	8	9	10	Total
/10	/20	/15	/15	/15	/15	/7	/8	/15	/15	/135

1. [10 Marks]

Circle the correct answer for each of the following statements. You will get one mark for each correct answer, you will get zero for each question left blank, and you will get $-\frac{1}{2}$ mark for each incorrect answer.

(a) **True** or **False**: A C compiler will recognize all syntax errors in a program.

(b) **True** or **False**: The value of the expression `(int) 1.8*0.6` is 1

(c) **True** or **False**: The statement `i *= j + 2;` is equivalent to `i = i*j + 2;`

(d) **True** or **False**: The operator < has higher precedence than the operator `||`

(e) **True** or **False**: Whenever a `for` loop is executed, the statement in the body of the loop is always executed at least once.

(f) **True** or **False**: In a function call, the type of the argument does not have to be identical to the type of the parameter.

(g) **True** or **False**: The value of the expression `strcmp("star","start")` is negative.

(h) **True** or **False**: If `s` is a pointer to a string whose value is `"sample"`, then the statement `printf("%c",list+2);` will print `m`.

(i) **True** or **False**: In order to be able to perform a binary search on a list, the items must be arranged in order.

(j) **True** or **False**: Recursive functions are usually more efficient than non-recursive ones.

2. [20 Marks]

Each part of this question is worth two marks.

(a) Write $\sin\sqrt{|x|}$ as a C expression.

(b) Write a statement that will assign to the `int` variable `choice` a random value from the set $\{10, 15, 20, 25, 30\}$.

(c) Write the simplest possible expression that is equivalent to the expression

```
!(a >= b || b >= c)
```

(d) Write 2103_4 as a base 10 numeral.

(e) Write 283 as a base 4 numeral.

(f) Trace a binary search as it seeks the value 80 in the array called `list` shown below. To show your trace, print the value of the array element examined at each stage of the search.

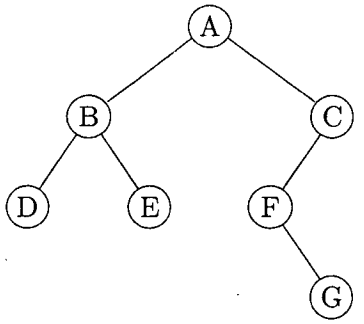
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`list`	10	18	20	27	34	45	52	60	68	70	78	82	88	90	95

(g) Suppose that an array initially contains the values `{8,2,4,7,3}`. If the array is to be sorted into ascending order using insertion sort, show the contents of the array after each of the first two passes of the sort.

(h) Suppose that an array initially contains the values `{5,3,8,4,2}`. If the array is to be sorted into ascending order using selection sort, show the contents of the array after each of the first two passes of the sort.

(i) Suppose that an array initially contains the values `{54,60,17,40,87,20,72}`. If the array is to be sorted into ascending order using quicksort as discussed in class, show the contents of the array after the first pass of the sort.

(j) The diagram shows a binary tree. In what order would the nodes be visited by a postorder traversal?



3. [15 Marks]

The number 153 has the property that it is equal to the sum of the cubes of its digits: $1^3+5^3+3^3 = 153$. Write a complete program that will find and print all the three-digit natural numbers that have this property.

4. [15 Marks]

Write the definition of a function `leastFactor` that has one `int` parameter, `n`. If `n` > 1, the function should return the value of the smallest prime factor of `n`; otherwise, it should return the value zero. (A prime number is a positive integer with exactly two factors: one and the number itself.)

5. [15 Marks]

Complete the definition of the function `closest` whose header is shown below. The function should return the smallest non-negative difference between values in the `int` array `list`. The number of items in the array is given by `listLength`. If the array has fewer than two values, the method should return zero.

As examples,

if `list = {3,13,20,10}` the function should return the value 3 (13 and 10 are closest)
if `list = {-4,5,2,-8,7,-1}` the function should return the value 2 (5 and 7 are closest)
if `list = {-5,7,-5,4}` the function should return the value 0 (-5 and -5 are closest)

```
int closest (int list[], int listLength)
```

6. [15 Marks]

Complete the definition of the function `averageWordLength` so that it returns, as a `double` value, the average length of the words in the string `s`. Assume that `s` consists only of words separated by one or more blanks with no leading or trailing blanks. You may also assume that there is at least one word in the string.

```
double averageWordLength (char *s)
```

7. [7 Marks]

Consider the recursive method whose definition is:

```
int f (int m, int n)
{
  int result;
  if (n == 0)
    result = 0;
  else
    result = m + f(m,n-1);
  return result;
}
```

(a) Evaluate `f(3,2)`.

(b) Evaluate `f(2,4)`.

(c) State in a few words (no more than ten) what `f(m,n)` determines for arbitrary values of `m` and `n`.

8. [8 Marks]

Write a recursive function that returns the number of occurrences of a character `c` in a string `s`. The function should have the header: `int count (char *s, char c)`

As examples,

`count("Toronto",'t')` should return 1
`count("Toronto",'o')` should return 3

Note that your function *must* be recursive. No credit will be given for a non-recursive solution.

9. [15 Marks]

Suppose that linked lists are implemented in the usual way seen in class with node structures defined as follows:

```
typedef struct node
{
  int info;
  struct node *link;
}Node, *NodePointer;
```

Complete the definition of the function `deleteLessThan` whose header is shown below. The function should delete all nodes whose `info` values are less than `item` from the list whose head is pointed to by the parameter `headPointer`.

```
void deleteLessThan (int item, NodePointer *headPointer)
```

10. [15 Marks]

Suppose that binary trees are implemented in the usual way seen in class with node structures defined as follows:

```
typedef struct node
{
  int info;
  struct node *lChild, *rChild;
}Node, *NodePointer;
```

If a binary tree is a binary search tree, this makes it easy to search for an item in the tree. It is still possible, however, to search for an item in a binary tree that is *not* organized as a binary search tree. Complete the definition of the function `isInTree` whose header is shown below so that it returns the value `true` if and only if `item` is in the binary tree. Your function should *not* assume that the tree is a binary search tree.

```
bool isInTree (int item, NodePointer root)
```

Extra Space *Please specify which question(s) you are answering on this page.*