UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS105F — Computer Fundamentals
Final Examination — December, 2006

Examiners: Tom Fairgrieve, W. James MacLean

Duration: 2.5 hours

- Examination Type A: This is a "closed book" examination; no aids are permitted.
- Calculator Type 4: No electronic or mechanical computing devices are permitted.
- Write your answers in the spaces provided. Please answer using a pen or dark pencil.
- If more space is required, blank pages are provided at the back of the examination. DO NOT detach any pages from the exam paper.
- Rough work, if necessary, can be done on the backs of the pages.
- This examination has 15 pages (including the cover page).
- Read all instructions and write your name, student number, and ECF login on the first page **before** you start.
- You must use the C++ programming language to answer programming questions.
- While you are not required to comment any code you write in this exam, useful comments *may improve your mark* if it helps the marker better understand what you are attempting to do.
- Except for the true/false questions, we will award partial marks where possible. Attempt answers to all questions.
- Enjoy the holiday season!

Last Name ______________________ First Name ______________________

Student Number ______________________ ECF Login ______________________

MARKS

Question	1	2	3	4	5	6	7	8	**Total**
Value	14	16	10	10	10	10	10	10	90
Mark									

1. (14 Marks, 1 mark each)
For (a) through (m), circle the correct answer for each of the following:

(a) **True or False:** An sorted linked list can be searched as quickly as a sorted array.

(b) **True or False:** When applied on an unsorted array, the insertion sort algorithm performs more element moves than selection sort.

(c) **True or False:** It is always safe to dereference a pointer variable.

(d) **True or False:** If we allocate a dynamic array and store the array's address in `arr`, then the statement `delete arr;` de-allocates the array.

(e) **True or False:** Adding a node at the head of a linked list, as described in class, is generally faster than adding it at the tail.

(f) **True or False:** Recursive algorithms may have more than one base case.

(g) **True or False:** Mergesort is faster than selection sort for large arrays.

(h) **True or False:** Structures may not be returned by functions.

(i) **True or False:** Structures may not have members that are themselves structures.

(j) **True or False:** The UNIX I/O redirection operator > appends to the end of a file, if it already exists.

(k) **True or False:** Absolute pathnames in UNIX may begin with any letter.

(l) **True or False:** A C++ function may return a variable of type pointer to `char`.

(m) **True or False:** When a C++ program ends, all of its dynamically allocated memory is returned to the operating system.

(n) If `i` is a variable and `p` points to `i`, which of the following expressions are aliases for `i`?

(1) `*p`	(2) `*&p`	(3) `*i`	(4) `*&i`
(5) `&p`	(6) `&*p`	(7) `&i`	(8) `&*i`

Circle all expressions that you think are aliases for `i`.

2. (16 Marks)

(a) [5 Marks] Write statements to count the number of characters in a file named `someData.txt`. Indicate any include directives your statements will require.

(b) [3 Marks] What is the output from the following program when executed with the input:
`Fee Fi Fo Fum`

```
#include <iostream>
using namespace std;
int main()
{
    char word[25], *data[4];

    for ( int i=0; i<4; i++ )
        cin >> word;
        data[i] = new char[25];
        data[i] = word;
    }

    for ( int i=0; i<4; i++ )
    {
        cout << data[i] << " ";
    }
    cout << endl;

    return 0;
}
```

(c) i. [2 Marks] State one reason for choosing to use an array instead of a linked list. Explain.

ii. [2 Marks] State one reason for choosing to use a linked list instead of an array. Explain.

(d) [4 Marks]
Write the function body for the function having declaration:

```
int *find_middle( int a[], int n );
```

When passed an array `int` a of length n, the function will return a pointer to the array's middle element. If n is even, choose the middle element with the larger index; for example, if `n = 4`, consider the middle element to be `a[2]`, not `a[1]`.

3. (10 Marks)

Write a C++ function named `findAll` to find **all** occurences of a value in an array of integers. The function will take, as parameters, the array, its size, and the value to be found. You will return a pointer to a linked list node of type

```
struct IndexNode
{
  int         index ;
  IndexNode *next   ;
};
```

If the array does not contain the value searched for, return `NULL`. Otherwise, return a pointer to the head node of a linked list. The list contains, in increasing order, the indices of all occurences of the value sought. **Note:** In order to get full marks, your list must contain the indices in increasing order. However, if you cannot achieve this, you may return them in reverse order at a penalty of 1 mark.

4. (10 Marks)

Write a function, having the header `void reverse(ListNode *&head)`, that reverses the order of the nodes in the linked list pointed to by the actual parameter corresponding to `head`. Do not make copies of the list, any of its nodes, or any of the `data` values. Assume `ListNode` is defined as

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

Hint: Think carefully before you start. There is a relatively short solution.

5. (10 Marks)

A **magic square** is an arrangement of the numbers from 1 to n^2 in an $n \times n$ array, with each number occurring exactly once, and such that the sum of the entries of any row, any column, and either diagonal are the same. It is not hard to show that this sum must be $\frac{n(n^2+1)}{2}$. The number n is called the **order** of the magic square.

For example, the 3×3 array

$$\begin{bmatrix} 6 & 7 & 2 \\ 1 & 5 & 9 \\ 8 & 3 & 4 \end{bmatrix}$$

is a magic square because it satisfies the properties listed above.

(a) [2 Marks] Assume there exists a library named `MatrixTrivia` that contains a function named `isMagicSquare`. `isMagicSquare` returns type `bool`. This function determines whether or not a dynamically allocated $n \times n$ array of `int` is a magic square. Write what you believe the function declaration should be for the `isMagicSquare` function. You are **not** required to write the body of the `isMagicSquare` function.

(b) [8 Marks] Write an efficient and complete C++ program that:

- Prompts the user to enter an order n.
- Allocates the amount of memory required to hold an $n \times n$ array of `int` variables.
- Reads an $n \times n$ array of integers into the allocated memory.
- Prints a message that indicates whether or not the input array is a magic square. Your program should use the `isMagicSquare` function discussed above. You are **not** required to write the body of the `isMagicSquare` function.
- Returns the allocated memory to the operating system.

You may assume that the user will enter a positive order n.

Continue your solution for Question 5 on this page.

6. (10 Marks) Recursion

(a) [5 Marks] What is the output of the printInOrder function, shown below, if it is called as

```
int arr[] = {5, 2, 3,-6, 2, 7};
printInOrder(arr,6);
```

You may find that it assists you in tracing the code if you draw a sketch (on the back of the previous page) showing the function calls and their parameters.

(Answer goes here)

```
void printBetween(const int b[], const int length,
                  const int index,
                  const int low, const int high)
{
  if (index != length)
  {
    if (b[index] > low && b[index] < high)
    {
      cout << "# " ; // extra debugging/tracing output
      printBetween(b,length,index+1,low,b[index]);
      cout << b[index] << " " ;
      printBetween(b,length,index+1,b[index], high);
    }
    else
    {
      cout << "* " ; // extra debugging/tracing output
      printBetween(b,length,index+1,low,high);
    }
  }
}

void printInOrder(const int a[], const int length)
{
  const int smallestInt = -2147483647;
  const int largestInt = 2147483647;
  printBetween(a,length,0, smallestInt,largestInt);
  cout << endl ;
}
```

(b) [1 Mark] If the purpose of the function is to print the elements of an array in non-decreasing order (even if it is not already sorted, and ignoring the '*' and '#' characters), is it working properly? If not, what is wrong? You may assume the input array will never contain the values specified by `smallestInt` and `largestInt`.

(c) [4 Marks] Modify the code to print the list in reverse (non-increasing) order. If you indicated in (b) that the code was not working properly, fix it as well. You do not need to rewrite the code in full if you can clearly explain what changes are required ... feel free to mark your changes directly on the code, which has been reproduced below (without the extra debugging/tracing output).

```
void printBetween(const int b[], const int length,
                  const int index,
                  const int low, const int high)
{
  if (index != length)
  {
    if (b[index] > low && b[index] < high)
    {
      printBetween(b,length,index+1,low,b[index]);
      cout << b[index] << " " ;
      printBetween(b,length,index+1,b[index], high);
    }
    else
    {
      printBetween(b,length,index+1,low,high);
    }
  }
}

void printInOrder(const int a[], const int length)
{
  const int smallestInt = -2147483647;
  const int largestInt = 2147483647;
  printBetween(a,length,0, smallestInt,largestInt);
  cout << endl ;
}
```

7. (10 Marks)

In this question you are to write a complete and efficient C++ function named findInString.

The findInString function locates the first complete occurrence of a given C-string s2 (excluding the terminating null character) in a given C-string s1. The index of the start of the string s2 in s1 is returned by findInString. The value -1 is returned if the string s2 is not found in the string s1. If s2 is the empty string, "", the findInString function returns the value 0. For example, findInString("splatter","latte") would return 2, while findInString("James","Tom") would return -1. You may call strlen, but no other functions, from your implementation of findInString.

8. (10 Marks)

An array of structure variables could be used to keep track of student records. Suppose the following statements were executed to set up such a list:

```
struct Student
{
    char *familyName;
    char *firstName;
    int  id;
};

Student list[400];
```

Lists of this nature are usually stored in a sorted alphabetical order by non-decreasing `familyName`. For example, the record for student "Fairgrieve" should appear in the list before the record for student "MacLean". If two or more students have the same `familyName`, the sorting can be based on non-decreasing `firstName`. And if two or more students have the same `familyName` and `firstName`, the sorting can be based on increasing `id` number.

(a) [5 Marks] Write an efficient C++ function named `isSorted` that determines both whether or not a given array of `Student` structure variables is in the order specified above.

(b) [5 Marks] Write an efficient C++ function that uses the sequential search algorithm to determine the first and last index of a given `familyName` in a sorted array of `Student` structures. If the given `familyName` does not appear in the array, both indices should be set to `-1`.

Continue answers to questions here if necessary.
Clearly indicate which question the answer belongs to.

Continue answers to questions here if necessary.
Clearly indicate which question the answer belongs to.