UNIVERSITY OF TORONTO

FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, DECEMBER 2004

First Year – Engineering Science

CSC 180H1F – INTRODUCTION TO COMPUTER PROGRAMMING

Exam Type: C

Examiners – M. Coahran and T.F. Fairgrieve

Aid sheet must be hand-written.

No calculators.

Student Number: ____________

Last Name: ____________

First Name: ____________

# 1:	______/ 4
# 2:	______/15
# 3:	______/ 7
# 4:	______/ 9
# 5:	______/ 6
# 6:	______/ 6
# 7:	______/10
# 8:	______/10
# 9:	______/13
TOTAL:	______/80

This final examination consists of 9 questions on 14 pages (including this one). When you are told to start, please make sure that your copy of the examination is complete. Answer each question directly on the examination paper, or on the back of a page if necessary.

This exam is SINGLE-sided.

Question 1. [4 MARKS]

Consider the two C functions named `findHist1()` and `findHist2()` given below. Assume that the array `image` holds `n` values each of which lies in the range `[0, MAXVAL]`, and that the array `hist` is of size `MAXVAL + 1`.

```
#include <stdio.h>
#define MAXVAL 255
void findHist1( int image[], int n, int hist[] ) {
    int i, j, opCount = 0;
    for ( i = 0; i <= MAXVAL; i++ ) {
        hist[i] = 0;
    }
    for ( i = 0; i <= MAXVAL; i++ ) {
        for ( j = 0; j < n; j++ ) {
            opCount++; /* ***********************    stmt A */
            if ( image[j] == i ) {
                hist[i]++;
            }
        }
    }
}

void findHist2( int image[], int n, int hist[] ) {
  int i, opCount = 0;
  for ( i = 0; i <= MAXVAL; i++ ) {
      hist[i] = 0;
  }
  for ( i = 0; i < n; i++ ) {
      opCount++; /* *****************************    stmt B */
      hist[image[i]]++;
  }
}
```

Part (a) [2 MARKS]

Give an expression for the number of times the statement labeled `stmt A` is executed. Justify your answer.

Part (b) [2 MARKS]

Give an expression for the number of times the statement labeled `stmt B` is executed. Justify your answer.

Question 2. [15 MARKS]

Part (a) [5 MARKS]

Consider the following C program:

```
#include <stdio.h>
int main() {

    int i, j, x=0;
    for ( i = 0; i < 4; i++ ) {
        for ( j = 0; j < i; j++ ) {
            x = x + (i + j - 1);
            printf("%d ", x );
        }
    }
    printf("\nx = %d\n", x);
    return 0;
}
```

Trace through this C program **carefully** and show your work. **Write** the output from an execution of this C program in the box at the bottom of this page.

output :

Part (b) [5 MARKS]

Consider the following C program:

```
#include <stdio.h>

void enigma( int n );

int main() {

    enigma( 13 );
    printf("\n");
    return 0;
}

void enigma( int n ) {

    if ( n != 0 ) {
        enigma( n/2 );
        printf("%d", n%2 );
    }
}
```

Trace through this C program **carefully** and show your work. **Write** the output from an execution of this C program in the box at the bottom of this page.

output :

Part (c) [5 MARKS]

Consider the following C program:

```
#include <stdio.h>
int main() {
    int a[] = {1,7,4,0,9,3,4,6,8};
    int *p, *q;
    int temp, i, n=9;

    p = a;
    q = a + n - 1;
    while ( p < q ) {
        while ( (p < q)  &&  (*p % 2 == 0) ) {
            p++;
        }
        while ( (p < q)  &&  (*q % 2 != 0) ) {
            q--;
        }

        temp = *p;
        *p = *q;
        *q = temp;

        for ( i = 0; i < n; i++ ) {
            printf("%d ", a[i]);
        }
        printf("\n");
    }
    return 0;
}
```

Trace through this C program **carefully** and show your work. **Write** the output from an execution of this C program in the box at the bottom of this page.

output :

Question 3. [7 MARKS]

In the English alphabet, each letter is classified as being either a vowel or a consonant. In English, the vowels are the letters **a**, **e**, **i**, **o**, **u** and **y**.

Part (a) [4 MARKS]

Write an efficient C function named `isVowel( char ch )` that returns 1 if `ch` is an English vowel and 0 otherwise. **Assume** that `ch` holds a **lower-case** letter from the English alphabet.
Write the function in the space below.

```
#define TRUE   1
#define FALSE  0
/* Function to determine whether or not the character ch is an English vowel. */
```

Part (b) [3 MARKS]

Write an efficient C function named `isConsonant( char ch )` that returns 1 if `ch` is an English consonant and 0 otherwise. **Assume** that `ch` holds a **lower-case** letter from the English alphabet.
Write the function in the space below.

```
#define TRUE   1
#define FALSE  0
/* Function to determine whether or not the character ch is an English consonant. */
```

Question 4. [9 MARKS]

Consider the following C program that is supposed to simulate a bank account. The program keeps track of the balance in a bank account by adding in the amount of a deposit, subtracting the amount of a cheque and charging a $10 service charge fee for each cheque that makes the account balance negative. Note that the program keeps track of only the number of **dollars** in the account.

```
#include <stdio.h>

int main() {

    int entry, balance;

    printf("This program simulates a bank account.\n");
    printf("Enter each cheque and deposit during the month.\n");
    printf("To indicate a cheque, enter a negative number.\n");
    printf("To indicate a deposit, enter a positive number.\n");
    printf("To quit, enter 0 for the cheque/deposit amount.\n\n");

    printf("Enter the initial balance: ");
    scanf("%d", &balance);

    printf("Enter amount of cheque (-) or deposit: ");
    scanf("%d", &entry);

    while ( entry != 0 ) {
        balance += entry;
        if ( balance < 0 ) {
            printf("This cheque bounces. $10 fee deducted.\n");
            balance -= 10;
        }

        printf("Current balance: %d\n", balance);

        printf("Enter amount of cheque (-) or deposit: ");
        scanf("%d", &entry);
    }

    printf("Final balance: %d\n", balance);

    return 0;
}
```

Part (a) [5 MARKS]

Describe different test cases that you would use to judge whether or not the bank account program works correctly.

	Description of test case input.	Reason for performing the test.
1		
2		
3		
4		
5		
6		
7		
8		

(You do not need to fill every entry in the above table to earn full marks.)

Part (b) [4 MARKS]

Does the bank account program work correctly? If not, what does it do wrong? Justify your response.

Question 5. [6 MARKS]

Variables having the following **struct** type may be used to represent time values that are given in an hours, minutes and seconds (hh:mm:ss.sss...) format.

```
typedef struct {
    int hours;
    int mins;
    double secs;
} timeinfo_t;
```

Members `mins` and `secs` should both contain values that are greater than or equal to 0 and less than 60.

A C function that adds time value variables `t1` and `t2`, and returns the sum of their time values might have the prototype:

```
timeinfo_t addTimes( timeinfo_t t1, timeinfo_t t2 );
```

When given arguments that represent the times 5:30:10.33 and 6:43:1.37, the `addTimes` function should return a variable that represents the time 12:13:11.7 (12 hours, 13 minutes, and 11.7 seconds). You may assume that the `mins` and `secs` members of the time value variables `t1` and `t2` contain values that are greater than or equal to 0 and less than 60.

Complete the `addTimes` function in the space provided below.

```
/* Function to determine the sum of two time values.  The type timeinfo_t   *
 * is used to represent time values that are given in an hours, minutes and *
 * seconds (hh:mm:ss.sss...) format.                                        */

timeinfo_t addTimes( timeinfo_t t1, timeinfo_t t2 ) {










}
```

Question 6. [6 MARKS]

A square, dimension n matrix **B** is called the **matrix transpose** of a square, dimension n matrix **A** if and only if the i, j element of **B** equals the j, i element of **A**, for all $i, j = 1, 2, 3, \ldots, n$.

Write C statements that **replace** the entries in a square, dimension n matrix **A** with its matrix transpose. Assume that a matrix is stored using a 2D array.

We have started a C program for you. Insert your C statements (including declarations) in the appropriate places. **Do not** declare any new **array** variables. Assume that two functions named `readMatrix()` and `printMatrix()` have been supplied to you separately. They are used to read and print matrices, respectively, using array elements `A[0][0]` through `A[n-1][n-1]`.

```
#include <stdio.h>
#define MAX_N 100
void readMatrix( double A[MAX_N][MAX_N], int n, int nMax );
void printMatrix( double A[MAX_N][MAX_N], int n );
int main() {

    int n;
    double A[MAX_N][MAX_N];
    /* ....  Declare any additional variables below. Remember, no arrays! .... */




    /* Read in the matrix A and its dimension. */
    readMatrix( A, n, MAX_N );

    /* Determine the matrix transpose of A. Store the result in A.   *
     * .... Write your C statements below ....                       */












    /* print the matrix transpose of A.  */
    printf(" A transpose = \n");
    printMatrix( A, n );
    return 0;
}
```

Question 7. [10 MARKS]

Write an efficient C function having the prototype `void pluralForm(char word[]);`. Your function should modify the given character string `word` from its singular form to its plural form using the following English grammar rules for nouns:

- If `word` ends in the letter(s) **s**, **x**, **z**, **ch**, or **sh**, add the letters **es** to the end of `word`.
- If `word` ends in the letter **y** and the **y** is preceded by a consonant, replace the letter **y** with the letters **ies**.
- In all other cases, add the letter **s** to the end of `word`.

Assume you already have a function named `isConsonant()` that takes a `char` argument and returns `TRUE` if the `char` is a consonant and returns `FALSE` otherwise. Your `pluralForm` function may call the function `isConsonant()`. You **do not** have to write the function `isConsonant()`.

Assume that the function `pluralForm` will always be passed the singular form of a noun, in lower-case. Also assume that the nouns entered will be short enough that modifying them in the prescribed manner will not overflow the space allocated for the string.

You may use any C library functions you know.

```
/* Function to modify the given character string word from its singular form *
 * to its plural form using English grammar rules.  It is assumed that the   *
 * variable word represents the singular form of a noun, in lower-case.     */

void pluralForm(char word[]) {






}
```

Question 8. [10 MARKS]

Spherical balls can be stacked to form a rectangular pyramid with one ball at the top, sitting on top of a square composed of four balls, sitting on top of a square composed of nine balls, sitting on top of a square composed of sixteen balls, and so forth. Write a **recursive** C function named `pyramid` that takes as its single argument the height of a rectangular pyramid and returns the total number of balls in a rectangular pyramid of the given height. Your function must have the prototype

```
int pyramid( int height );
```

You may **assume** your function will be passed a nonnegative number.

The height of the rectangular pyramid is measured in units of number of balls. For example, the rectangular pyramid with one ball at the top, sitting on top of a square of four balls has height 2.

You **must** write a recursive function. A nonrecursive solution will earn no marks.

Question 9. [13 MARKS]

A mathematical function P is called a polynomial if

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1}x^{n-1} + a_n x^n$$

where n is a nonnegative integer and the numbers a_0, a_1, a_2, ..., a_n are constants called the coefficients of the polynomial. If the coefficient a_n is nonzero, then the degree of the polynomial is n.

Write an efficient C function that, given the coefficients of a degree n polynomial P, determines the coefficients of the k^{th} **derivative** of P, for some nonnegative integer k. Make use of:

- the power rule: $\frac{d}{dx}(x^m) = mx^{m-1}$,
- the constant multiple rule: $\frac{d}{dx}(cf(x)) = c\frac{d}{dx}(f(x))$,
- and the sum rule: $\frac{d}{dx}(f(x) + g(x)) = \frac{d}{dx}(f(x)) + \frac{d}{dx}(g(x))$.

Here, m is a positive integer, c is a constant, and f and g are differentiable functions. In general, the k^{th} derivative of a function f is obtained from f by differentiating k times.

To make your task a little simpler, assume that your C function has two equal-length array parameters, one for holding the coefficients of P and the other for holding the coefficients of the k^{th} derivative of P. You get to specify the other parameters.

Assume that the polynomial coefficients are real-valued and may be stored in `double` variables. Assume also that the given value of k is nonnegative.

For example, your function should determine that the 2^{nd} derivative of the polynomial having coefficients

```
a[] = { 1.0, 2.0, 3.0, 4.0 }
```

has coefficients

```
b[] = { 6.0, 24.0, 0.0, 0.0 }.
```

HINT: Start out by writing down the first few derivatives of P. Then write your C function.

Use the space below for rough work. Place your Question 9 solution on the next page.

Please enter your Question 9 solution here.

Total Marks = 80