UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS 105 — Computer Fundamentals
Midterm Examination
March 1, 2018
6:10 p.m. – 8:00 p.m.
(110 minutes)

Examiners: B. Li, B. Korst, H. Shokrollah-Timorabadi and M. Stumm

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

Calculator Type 4: No calculators or other electronic devices are allowed.

All questions are to be answered on the examination paper. If the space provided for a question is insufficient, you may use the last page to complete your answer. If you use the last page, please direct the marker to that page and indicate clearly on that page which question(s) you are answering there.

You must use the C programming language to answer programming questions. You are not required write `#include` directives in your solutions. Except those excluded by specific questions, you may use functions from the `math` library as necessary.

The examination has 13 pages, including this one.

First Name: _____ Last Name: _____

Student Number: _____

MARKS

1	2	3	4	5	6	7	8	9	10	11	12	13	Total
/4	/4	/4	/4	/4	/8	/8	/8	/11	/11	/11	/11	/12	/100

Question 1 [4 Marks]

Write a single C statement — which contains exactly one terminating semi-colon (`';'`), and does not contain brace brackets (`'{'` or `'}'`) — that declares a `double` variable `result`, and initializes it with the result of computing the following mathematical formula, based on variables `x` and `y`:

$$\sqrt{\frac{3\sin x}{1-4e^{y}}}$$

You can assume that all of the functions in the C `math` library are available, `x` and `y` have already been declared as `double`-type variables, and `x` is a value in radians (rather than degrees).
Solution:

```
result = sqrt(3 * sin(x) / (1 - 4 * exp(y)));
```

Question 2 [4 Marks]

Write a single C statement — which contains exactly one terminating semi-colon (`';'`), and does not contain brace brackets (`'{'` or `'}'`) — that declares a `bool` variable named `isHighlighted`, and sets its value to `true` if and only if the value stored in an integer variable named `characterCount` is an even positive number. Assume that the variable `characterCount` has already been declared and initialized.

Solution:

```
bool isHighlighted = characterCount > 0 && characterCount % 2 == 0;
```

Question 3 [4 Marks]

In most companies, after you write some code, it gets reviewed by other programmers during a so-called "code review" before the code is ever used. The purpose of a code review is to make sure the code is well written, concise, understandable and correct.

Joe Wannabe Hacker, who works for FancyCo, submitted the following piece of C code for review:

```
int i = 0;

do {
  printf("Count: %d\n", i + 1);
  i++;
} while (i < 2);
```

This code was subsequently criticized when code-reviewed. Other programmers on the code review team insisted that the same code could be written in just a single C statement with the C programming language.

Your task is to show that the code review team was right. In the space below, please provide a single C statement that has the same outcome as the code submitted by Joe Wannabe Hacker. A

single C statement contains exactly one terminating semi-colon (‘;’), and does not contain brace brackets (‘{’ or ‘}’).

Solution:

```
printf("Count: 1\nCount: 2\n");
```

Question 4 [4 Marks]

Complete the following C function, called `mostSignificantDigit`, that returns the most significant digit of a positive `int`-type integer that is passed to the function.

For example, if the function is called with the value `987654321` as its argument, it will return `9`.

Solution:

```
#include <stdio.h>

int mostSignificantDigit(int number) {
    int leadingDigit;

    while (number > 0) {
        leadingDigit = number % 10;
        number /= 10;
    }

    return leadingDigit;
}
```

Question 5 [4 Marks]

The variable `numApples` is an `int` type variable representing the number of apples in a barrel. The owner of the apples is deciding whether to sell them in packages of 3 or 5 apples. Write a single C statement that declares and initializes an `int` type variable called `leftover`. `leftover` should be initialized to the minimum of two quantities: 1) the number of apples left over when the barrel of apples is packaged into groups of 3; 2) the number of apples left over when the barrel of apples is packaged into groups of 5.

Hint: Use a function in the `math` library.

Solution:

```
int leftover = fmin(numApples % 3, numApples % 5);
```

Question 6 [8 Marks]

Give the implementation of a C function called `findMaximumValue` that takes an `int` array `list` and its size, called `size`, as its two parameters, and returns the maximum `int` value in the array if it is positive. If the maximum value in the array is 0 or negative, the function returns 0.

Solution:

```
int findMaximumValue(int list[], int size) {
  int maximumValue = 0;
  for (int i = 0; i < size; i ++) {
    if (maximumValue < list[i]) {
      maximumValue = list[i];
    }
  }
  return maximumValue;
}
```

Question 7 [8 Marks]

What is the output of the following program:

```
#include <stdio.h>

int *confuse(int *x, int *y) {
    (*y)++;
    y = x;
    *y = 10;
    return (y);
}

int main(void) {
    int a = 6, b = 7;
    int *f = &b;

    f = confuse(&a, &b);
    (*f)++;

    printf("a = %d and b = %d\n", a, b);
    return 0;
}
```

Solution:

```
a = 11 and b = 8
```

Question 8 [8 Marks]

Identify and correct all compile-time errors you find in the C program below. Compile-time errors are **errors** — not **warnings** — that the compiler will report when compiling the program. Each line may or may not contain compile-time errors, and there may be more than one error per line.

```
1       #include <stdio.h>

2       int main(void) {
3            double a, b = 3.14;

4            do {
5              int i = 0;
6              printf("Enter a positive integer for offset: \n");
7              scanf("%d", &a);
8            } while (i < 5 && (a < 100 || a > 1);

9            int j;
10           for (j = 0, j < 3, j++) {
11                  y = b * j % a;
12                  printf("%d\n", y);
13           }
14           return 0;
15       }
```

Solution:

```
Line 5: should be declared outside of the loop for the conditional expression to be
evaluated

Line 8: Missing closing `)'

Line 10: Comma should be semi-colon

Line 11: Cannot use modulo operator with double values, should be corrected to
y = b * j % (int) a; (or declare variable 'a' as int)

Line 11: variable 'y' is not declared but used here
```

Question 9 [11 Marks]

The dot product, an operation with which every first-year engineer is familiar, consists of the element-by-element multiplication of two vectors, and the cumulative sum of these resulting products. If vector $a = [a_1\ a_2\ a_3]$, and $b = [b_1\ b_2\ b_3]$, then the dot product $a \cdot b = a_1 \times b_1 + a_2 \times b_2 + a_3 \times b_3$.

Smartphones, which can be found at this very moment in many first-year engineer's pocket, do perform a similar operation when the treble or the bass are adjusted when the said engineer is enjoying a song. This operation is called *filtering*.

Suppose now that you have two vectors, one representing a song, and the other representing a filter. Write a complete C program that will calculate and print the dot product between these two vectors as a single value. Your program should simply print:

```
Result = <calculated dot product value here>
```

Your program must use a function, called `dotProduct`, which takes in pointers to the two vectors and their length, and returns the result. Before your program calls `dotProduct`, the two vectors should be initialized using the following elements:

```
music: 0 0.707 1 0.707 0 -0.707 -1 -0.707 0    // it's a sinusoid
filter: 1 0 -1 0 2 0 -1 0 1                    // it's a sinc
```

Next time you listen to a song, consider that it is very possible that two 50 element long arrays are being used by a function very similar to the one you will write below, and that function is being called at least once every 48 thousandths of a second, so that you can enjoy that Taylor Swift song. Okay, make it Justin Bieber, then.

Solution:

```
#include <stdio.h>

// The function prototype can also be declared as:
// double dotProduct(double music[], double filter[], int length) {
double dotProduct(double *music, double *filter, int length) {
    double sum = 0.0;
    int i;

    for (i = 0; i < length; i++) {
        sum += music[i] * filter[i]; // or *(music + i) * *(filter + i)
    }

    return sum;
}

int main (void) {
    double music[9] = {0, 0.707, 1, 0.707, 0, -0.707, -1, -0.707, 0};
    double filter[9] = {1, 0, -1, 0, 2, 0, -1, 0, 1};
    printf("Result = %lf\n", dotProduct(music, filter, 9));
}
```

Question 10 [11 Marks]

Write a C function called `generateRandomPrimeNumber` that returns a randomly generated prime number between 1 and a maximum `int`-type integer, `maxRange` (inclusive), which is provided as an argument when calling the function, and is assumed to be greater than 1. A prime number is a

natural number greater than 1 that cannot be formed by multiplying two smaller natural numbers. For example, 2, 3, 5, 11, and 13 are all prime numbers.

You are not allowed to use arrays or pointer variables in your implementation. For convenience, you do not need to seed the random number generator.

Solution:

```
int generateRandomPrimeNumber(int maxRange) {
    int primeNum;
    bool foundNotPrime;

    do {
        foundNotPrime = false;
        primeNum = rand() % maxRange + 1;

        // check if primeNum is prime
        for(int i = 2; !foundNotPrime && i <= primeNum / 2; i++) {
            if (primeNum % i == 0) {
                foundNotPrime = true;
            }
        }

        // Check special cases: to prevent 1 of becoming prime
        if (primeNum == 1)
             foundNotPrime = true;
    } while (foundNotPrime);

    return primeNum;
}
```

Question 11 [11 Marks]

Recall that the function `rand()` returns a random integer each time it is called. Write a complete C program to help assess the quality of `rand()`, by following the three steps provided below.

First, declare an array with the identifier `random` that contains 1,000 `int`-type integers, and then fill this array with random numbers between `0` and `255` (inclusive).

Second, declare another array with the identifier `h` that contains 256 integers, and use that array to create a histogram so that at the end of the program, for each `i` between `0` and `255` (inclusive), `h[i]` will have a value `x` if exactly `x` elements of array `random` have the value `i`.

Finally, print out the values of all elements of `h`.

For the sake of convenience, you do not need to seed the random number generator.

(The quality of `rand()` can then be assessed by someone who uses your program as follows: if the printed-out numbers are all within a small range, then the quality of `rand()` is pretty good; on the other hand, if the printed-out numbers span a large range, then the quality of `rand()` is rather poor.)

Solution:

```
int main(void) {
    int random[1000];
    for (int i = 0; i < 1000; i++)
      random[i] = rand() % 256;

    int h[256];
    // must first initialize elements of h to 0
    // can also use int h[256] = {0}; (or {})
    for(int i = 0; i < 256; i++)
        h[i] = 0;
    // now build histogram
    for(int i = 0; i < 1000; i++)
        h[ random[i] ]++;

    for(int i = 0; i < 256; i++)
        printf("%d", h[i]);
    printf("\n");
    return 0;
 }
```

Question 12 [11 Marks]

Write a complete C program that prompts the user repeatedly for a sequence of up to 10 integer values. After receiving all 10 values, **or** if the user enters `0`, the program will stop prompting for more values. You can assume that the user enters at least one value before entering `0`. Your program will complete the following three tasks, in the order as given below, using the values the user entered:

- Print the total number of values entered;
- Print all the values in the order that the user entered them;
- Print whether the values are entered in ascending order, i.e., the next value is either greater than or equal to the previous one. For example, `{3, 4, 7, 7}` is a sequence of values in ascending order, but `{3, 4, 7, 6}` is not.

Hint: you will need to use an array for this question.

Here are a few example runs of your program.

Example run 1:

```
Enter a value (0 to stop): 1
Enter a value (0 to stop): 2
Enter a value (0 to stop): 3
Enter a value (0 to stop): 4
Enter a value (0 to stop): 7
Enter a value (0 to stop): 7
Enter a value (0 to stop): 8
Enter a value (0 to stop): 9
Enter a value (0 to stop): 10
Enter a value (0 to stop): 11
There are a total of 10 numbers.
The values you entered are: 1 2 3 4 7 7 8 9 10 11
The values are in ascending order.
```

Example run 2:

```
Enter a value (0 to stop): 3
Enter a value (0 to stop): 5
Enter a value (0 to stop): 5
Enter a value (0 to stop): 7
Enter a value (0 to stop): 0
There are a total of 4 numbers.
The values you entered are: 3 5 5 7
The values are in ascending order.
```

Example run 3:

```
Enter a value (0 to stop): 2
Enter a value (0 to stop): 1
Enter a value (0 to stop): 3
Enter a value (0 to stop): 0
There are a total of 3 numbers.
The values you entered are: 2 1 3
The values are not in ascending order.
```

Example run 4:

```
Enter a value (0 to stop): 1
Enter a value (0 to stop): 0
There are a total of 1 numbers.
The values you entered are: 1
The values are in ascending order.
```

Solution:

```c
#include <stdio.h>
#include <stdbool.h>

int main(void) {
  int listOfNumbers[10];
  int i = 0, value;
  int total = 0;
  bool ascending = true;

  do {
    printf("Enter a value (0 to stop): ");
    scanf("%d", &value);

    listOfNumbers[i] = value;
    i = i + 1;

    if (value != 0) {
      total = total + 1;
    }
  } while (i < 10 && value != 0);

  printf("There are a total of %d numbers.\n", total);

  printf("The values you entered are: ");
  for (int i = 0; i < 10 && listOfNumbers[i] != 0; i ++) {
    printf("%d ", listOfNumbers[i]);

    if (i < total - 1) {
      if (listOfNumbers[i] > listOfNumbers[i + 1]) {
        ascending = false;
      }
    }
  }

  printf("\n");

  if (ascending)
    printf("The values are in ascending order.\n");
  else
    printf("The values are not in ascending order.\n");

  return 0;
}
```

Question 13 [12 Marks]

The constant `E` is defined as a `double` constant of `2.718281828459045`.

```
const double E = 2.718281828459045;
```

A first positive integer is called a **mirror** of a second one if they both contain two digits, and when the two digits in the first integer are flipped, the first integer becomes the second one. For example, `81` is a mirror of `18` (and vice versa).

Implement a function called `firstMirrorInE` that returns the first two-digit number found in consecutive digits of `E` whose mirror have appeared earlier in the sequence of digits. You should only consider the first 16 digits of E — `2718281828459045`. The function returns `0` if such a mirror pair does not exist in the first 16 consecutive digits of `E`.

Hint: The `firstMirrorInE` function should return `28`, since its mirror, `82`, has appeared earlier in the sequence of digits. Your function must not simply return `28` without doing any work. It is also incorrect to return `81`, because even though its mirror, `18`, appeared previously, `81` is not the first in the sequence that can be found.

Feel free to declare and implement additional functions when needed.
Solution:

```
bool mirror(int i, int j) {
  int firstDigit = i / 10;
  int secondDigit = i % 10;
  return j == secondDigit * 10 + firstDigit;
}

int firstMirrorInE(void) {
  const double E = 2.718281828459045;
  const int NumberOfDigits = 15;
  int count = 0;
  int twoDigitNumbers[17] = {0};

  for (int i = 0; i >= -NumberOfDigits; i --)
  {
     int p1 = (int) (E / pow(10, i)) % 10;
     int p2 = (int) (E / pow(10, i - 1)) % 10;

     int p = p1 * 10 + p2;

     for (int j = 0; j < count; j ++) {
       if (mirror(twoDigitNumbers[j], p))
         return p;
     }

     twoDigitNumbers[count] = p;
```

```
     count++;
  }
  return 0;
}

Alternative solution:

int firstMirrorInE(void)
{
  const double E = 2.718281828459045;
  int count = 0;
  int twoDigitNumbers[17] = {0};

  double e = E;
  int p, currentDigit, previousDigit = -1;

  for (int i = 1; i <= 16; i++)
  {
     currentDigit = (int) e;

     e = (e - (int) e) * 10;
     if (previousDigit != -1)
     {
        p = currentDigit + previousDigit * 10;

        for (int j = 0; j < count; j ++) {
           if (mirror(twoDigitNumbers[j], p))
             return p;
        }

        twoDigitNumbers[count] = p;
        count++;
     }
     previousDigit = currentDigit;
  }
  return 0;
}
```

This page has been left blank intentionally. You may use it for answers to any question in this examination.