

**UNIVERSITY OF TORONTO**  
**FACULTY OF APPLIED SCIENCE AND ENGINEERING**

**APS 105 — Computer Fundamentals**  
**Midterm Examination**  
**October 18, 2012**  
**6:15 p.m. – 8:00 p.m.**  
**(105 minutes)**

**Examiners: J. Anderson, B. Li, M. Sadoghi, D. Sengupta, G. Steffan**

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. You may use any math function that you have learned, as necessary.

The examination has 16 pages, including this one.

Circle your lecture section (**one mark deduction** if you do not correctly indicate your section):

L01	L02	L03	L04	L05	L06
Li	Li	Anderson	Steffan	Sengupta	Sadoghi
Monday 2 PM	Monday 9 AM	Monday 11 AM	Monday 11 AM	Monday 4 PM	Monday 4 PM

Full Name: \_\_\_\_\_

Student Number: \_\_\_\_\_ UTORID: \_\_\_\_\_

**MARKS**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
/4	/4	/4	/4	/4	/5	/5	/5	/5	/12	/12	/12	/12	/12	/100

**Question 1 [4 Marks]**

Supposing that A, B, C and D are `bool` type variables, write a single C statement that is equivalent to the following statement, but that does not use the `&&` Boolean operator.

```
bool F = (A || B) && (C || D);
```

**Solution:**

```
bool F = !( !(A || B) || !(C || D));
```

**Question 2 [4 Marks]**

Given that `int` type variables  $x_1, y_1$  represent a point in the Cartesian plane, and that `int` type variables  $x_2, y_2$  represent a second point in the Cartesian plane. In the Cartesian plane, suppose that these two points form a line. Write a single C statement that declares a `double` type variable b and initializes b to the intercept of the line with the y axis. (Hint: the intercept of a line formed by  $(x_1, y_1)$  and  $(x_2, y_2)$  with the y axis can be computed mathematically as  $y_1 - \frac{y_2 - y_1}{x_2 - x_1} \cdot x_1$ .)

**Solution:**

```
double b = y1 - ((double)(y2-y1)/(x2-x1))*x1;
```

**Question 3 [4 Marks]**

Write a single C statement that declares an `int` type variable named q and initializes it to a random number between 0 and 201 inclusive, where the random number is divisible by 3.

**Solution:**

```
int q = (rand() % 68) * 3;
```

**Question 4 [4 Marks]**

Suppose that i is an `int` variable with value 8, and j is a `double` variable with value 2.2. Please evaluate the following expression (to `true` or `false`).

**Solution:**

Expression	Answer
<code>(i == (int)(j * 4))</code>	<code>true</code>
<code>(i == ((int)j) * 4)</code>	<code>true</code>
<code>(i == j * 4)</code>	<code>false</code>
<code>(!i)</code>	<code>false</code>

### **Question 5 [4 Marks]**

Circle and fix the errors in the following complete C program:

```
#include <stdio.h>

const double SPECIAL_CONST = 5.6

int main(void)
{
    printf("Enter a value:\n");
    double q;
    scanf("%l", q);
    printf("The adjusted value is: %f\n", q*SPECIAL_CONST)
    return 0;
}
```

After all the errors have been fixed, the output of the program is as follows:

```
Enter a value:
2.5 <enter>
The adjusted value is: 14.000000
```

### **Solution:**

- Missing semicolon at the end of the constant declaration.
- Should be %lf in the scanf.
- Should be &q in the scanf.
- Missing semicolon at the end of the second printf.

**Question 6 [5 Marks]**

Given the following C program:

```
#include <stdio.h>

int main(void)
{
    int row, col;
    const int MAX = 5;

    for (row = 0; row < MAX; row++)
    {
        for (col = 1; col < MAX - row; col++)
            printf("--");

        for (col = 1; col <= row; col++)
            if (row % 2 == 0)
                printf("%d-", col);
            else
                printf("%d-", 5 + col);
        printf("%d-", MAX);

        printf("\n");
    }
    return 0;
}
```

What is the output from an execution of this C program?

**Solution:**

```
-----5-
-----6-5-
----1-2-5-
--6-7-8-5-
1-2-3-4-5-
```

**Question 7 [5 Marks]**

What will be printed by the following C program?

```
int myFunc(int a, int b)
{
    int x;
    do
    {
        x = a;
        a = b % a;
        b = x;
        printf("a:%d ", a);
    } while (!(a==0));
    return b;
}

int main(void)
{
    printf("%d\n", myFunc(32, 23));
}
```

**Solution:**

a:23 a:9 a:5 a:4 a:1 a:0 1

**Question 8 [5 Marks]**

What will be printed by the following C program?

```
int main(void)
{
    char letter = 'a';
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j <= i; j++)
        {
            printf("%c", letter);
            letter++;
        }
        printf("\n");
    }
    return 0;
}
```

**Solution:**

a  
bc  
def  
ghij

**Question 9 [5 Marks]**

For each of the following parts, determine the number of times that the word Engineering will be printed. Place your answer in the appropriate box.

```
(a)    for (int i = 0; i < 100; i++)
{
    while (i % 10 >= 5)
    {
        printf("Engineering\n");
        i++;
    }
}
```

**Solution:**

Will print Engineering 50 times.

```
(b)    for (int i = 0; i < 5; i++)
        for (int j = i; j < 10; j++)
            printf("Engineering\n");
```

**Solution:**

Will print Engineering 40 times.

### Question 10 [12 Marks]

Give the implementation of a function called `bothMinAndMax` that computes *both* the minimum and maximum values of three integer parameters `x`, `y`, and `z`, and returns the minimum and maximum values via the pointer parameters `minPtr` and `maxPtr`.

**Note 1:** you must also add a proper call to `bothMinAndMax` in the `main` function below, in the position indicated, such that `bothMinAndMax` will modify `smallest` to contain the minimum of `a`, `b`, and `c`, and will modify `largest` to contain the maximum of `a`, `b`, and `c`.

For the values of `a`, `b`, and `c` given below, `main` will print: Of 7, 5, and 9, 5 is the smallest and 9 is the largest.

**Note 2:** your implementation of `bothMinAndMax` should work for any input values of `x`, `y`, and `z`, not just the example values given below.

**Note 3:** when two variables have the same value, you can return either one; e.g., if `x` has the same value as `y`, and both are less than `z`, you can return either `x` or `y` as the minimum value.

```
#include <stdio.h>

void bothMinAndMax(int x, int y, int z, int *minPtr, int *maxPtr);

int main(void)
{
    int a = 7, b = 5, c = 9;
    int smallest, largest;

    // add your call to bothMinAndMax here:

    printf("Of %d, %d, and %d, %d is the smallest and %d is the largest.\n",
           a, b, c, smallest, largest);
    return 0;
}

// give the remainder of the implementation of bothMinAndMax here:
void bothMinAndMax(int x, int y, int z, int *minPtr, int *maxPtr)
```

### Solution:

```
void bothMinAndMax(int x, int y, int z, int *minPtr, int *maxPtr);

int main(void)
{
    int a = 7, b = 5, c = 9;
    int smallest, largest;
    bothMinAndMax(a, b, c, &smallest, &largest);
```

```

printf("Of %d, %d, and %d, %d is the smallest and %d is the largest.\n",
a, b, c, smallest, largest);
return 0;
}

void bothMinAndMax(int x, int y, int z, int *minPtr, int *maxPtr)
{
if (x > y)
{
    if (x > z)
        *maxPtr = x;
    else
        *maxPtr = z;
    if (y < z)
        *minPtr = y;
    else
        *minPtr = z;
}
else
{
    if (y > z)
        *maxPtr = y;
    else
        *maxPtr = z;
    if (z < x)
        *minPtr = z;
    else
        *minPtr = x;
}
}

```

### Question 11 [12 Marks]

Write a *complete* C program that guesses a user's secret number, denoted by  $s$ , by successively prompting the user to answer yes/no questions that narrow the range of possible values for the secret number. The program first prompts the user to enter a range of values  $[min, max]$  such that  $min \leq s \leq max$ . The program starts by asking the user

"Is your secret number greater than  $min + (max - min)/2$ ?"

If the user answers "yes", then we know that the user's secret number lies in the range  $[min + \frac{max - min}{2} + 1, max]$ ; otherwise the number lies in  $[min, min + \frac{max - min}{2}]$ . Therefore, after each round of questioning, the values of min and max are further refined (the range of values is divided by half).

The program continues to ask the question (after refining the min and max values based on user's response) until the value of min is equal to max. When the value of  $min = max$ , then the program has found the user's secret number, which is equal to  $s = min = max$ , and the program prints user's secret value and terminates.

Assume all user inputs including user's secret numbers are valid integers and  $min \leq max$ . Also assume that the user enters "1" for "yes" and "0" for "no".

The following is an example output:

```
Please enter the range of values for your secret number [min, max] :  
1 10 <enter>  
Is your secret number greater than 5  
1 <enter>  
Is your secret number greater than 8  
0 <enter>  
Is your secret number greater than 7  
0 <enter>  
Is your secret number greater than 6  
1 <enter>  
Your secret number is 7
```

### Solution:

```
int main(void)  
{  
    int min = 0, max = 0;  
    int answer;  
  
    //Prompting the user for a range of values and storing them properly [1 mark]  
    printf("Please enter the a range of values for your secret number:\n");  
    scanf("%d %d", &min, &max);  
  
    // Repeatedly asking yes/no questions using a loop [1 mark]  
    // Having a correct while loop condition (also min < max is acceptable) [1 mark]  
    while (min != max)  
    {  
        printf("Is your secret number greater than %d\n", (min + (max - min)/2));
```

```
scanf("%d", &answer);

//Cutting space in half depending on user response [1 mark]
//Handling cases where user's secret value is an odd number (e.g., adding +1 to min) [1 mark]
if(answer)
    min = (min + (max - min)/2) + 1;
else
    max = (min + (max - min)/2);
}

printf("Your secret number is %d\n", min);
return 0;
}
```

### Question 12 [12 Marks]

Write a *complete* C program that performs the following tasks.

- Asks the user for a non-zero positive integer and stores it in the variable `num`;
- Prints the numbers between 1 and `num` (inclusive), but
  - For multiples of three prints “Bizz” instead of the number.
  - For multiples of five prints “Buzz” instead of the number.
  - For multiples of both three and five prints “BizzBuzz” instead of the number.
- prints the sum of all the numbers between 1 and `num` not including Bizz, Buzz, and BizzBuzz (*i.e.*, the sum of all the non-Bizz, non-Buzz and non-BizzBuzz numbers).

You can assume that the user provides the correct input, a positive integer that fits within the range of an `int` variable (*i.e.* You do not have to explicitly check the sign and range of the integer). You are not allowed to use any other header file other than `stdio.h`. The following is an example run of the program with the data entered by the user displayed in **bold** font.

```
Enter number: 21<enter>
The BizzBuzz results are:
1
2
Bizz
4
Buzz
Bizz
7
8
Bizz
Buzz
11
Bizz
13
14
BizzBuzz
16
17
Bizz
19
Buzz
Bizz
sum is 112
```

**Solution:**

```
#include<stdio.h>

int main(void)
{
    int num, i;
    int sum = 0;

    printf("Enter number: ");
    scanf("%d", &num);

    printf("The BizzBuzz results are:\n");

    for(i = 1; i <= num; i++)
    {
        if (!(i%3))
            if (!(i%5))
                printf("BizzBuzz\n");
            else
                printf("Bizz\n");

        else if (!(i%5))
            printf("Buzz\n");
        else
        {
            printf("%d\n", i);
            sum += i;
        }
    }
    printf("sum is %d\n", sum);
    return 0;
}
```

**Question 13 [12 Marks]**

Write a complete C function with the following prototype:

```
int replaceDigits(int n, int d, int r);
```

that finds all occurrences of the integer digit  $d$  within the integer  $n$ , replaces the matching digits with the integer digit  $r$ , and then returns the resulting integer.

For example, if `replaceDigits(2565, 5, 3)` is called, the function will return the integer 2363.

You may assume that  $d$  and  $r$  are single integer digits in the range of 1 to 9 inclusive. You may not use arrays in your solution.

**Solution:**

```
int replaceDigits(int n, int d, int r)
{
    int newNum = 0;
    int multiplier = 1;
    while (n != 0) // while (n)
    {
        int extractDigit = n % 10;

        if (extractDigit == d)
            newNum += r * multiplier;
        else
            newNum += extractDigit * multiplier;

        multiplier *= 10;
        n /= 10;
    }
    return newNum;
}
```

### Question 14 [12 Marks]

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

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

Implement the firstPrimeInE function that returns the first two-digit prime number found in consecutive digits of E (considering the first 16 digits of E — 2718281828459045). It returns 0 if such a two-digit prime number does not exist in the first 16 consecutive digits of E, as defined above.

**Hint:** The firstPrimeInE function returns 71, since 27 is not a prime number, and 71 is. Your function must find the first two-digit prime in E by analyzing the digits within E. That is, your program may not simply return 71.

**Solution:**

```
int firstPrimeInE(void)
{
    bool found = false;
    const int NumberOfDigits = 15;

    for (int i = 0; !found && 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;

        found = true;
        for (int j = 2; j < p / 2; j++) // sqrt(p) and p are fine as well
            if (p % j == 0)
                found = false;

        if (found)
            return p;
    }
    return 0;
}
```

Alternative solution:

```
int firstPrimeInE(void)
{
    double e = E;
    int p, currentDigit, previousDigit = -1;
    bool found;

    for (int i = 1; i <= 16; i++)

```

```
{  
    currentDigit = (int) e;  
  
    e = (e - (int) e) * 10;  
    if (previousDigit != -1)  
    {  
        p = currentDigit + previousDigit * 10;  
  
        found = true;  
        for (int j = 2; j < p / 2; j++) // sqrt(p) and p are fine as well  
        if (p % j == 0)  
            found = false;  
        if (found)  
            return p;  
    }  
    previousDigit = currentDigit;  
}  
return 0;  
}
```

*This page has been left blank intentionally. You may use it for answers to any questions.*