CS1010E Programming Methodology

Semester 1 2016/2017

Week of 26 – 30 September 2016

Tutorial 5

Value-Returning Functions

1. In tutorial #1, we performed a trace of a program containing only the `main` function. Extend your *mental model* to include traces of program execution involving more than one function using the program below.

```
#include <stdio.h>

int f(int x, int y);

int main(void) {
   int x = 3, y = 4;

   x = f(x,y);
   y = f(x, f(y,x));
   printf("x = %d; y = %d\n", x, y);
   return 0;
}

int f(int x, int y) {
   return x*10 + y;
}
```

   Keep in mind the following notions while you trace.

   - function call with evaluated arguments
   - function activation with parameter declaration
   - pass-by-value
   - lexical scoping
   - function termination upon return

2. In the following parts, you are required to define functions to find the maximum of a set of integer values. *Note that some of the values might be the same.* Devise your own `main` function to test the correctness of each of the functions.

   (a) Define a function `max2` that takes in two values a and b, and returns the maximum of the two values.

   ```
   int max2(int a, int b);
   ```

   (b) Define a function `max3` that takes in three values a, b and c, and returns the maximum of the three values. Use the `max2` function defined in part 2a above.

   ```
   int max3(int a, int b, int c);
   ```

3. The following are some guiding principles when defining functions.

   - A function should be reusable.
   - A function should perform a single well-defined task.
   - A function should rely on minimal input to do its work.

   Determine whether the following functions meet the above criteria.

   (a)
   ```
   double areaCircle(void) {
       double radius, area;

       printf("Please enter radius of circle: ");
       scanf("%lf", &radius );

       area = 3.14159 * radius * radius;

       return area;
   }
   ```

   (b)
   ```
   double areaCircle(double radius, double area) {
       area = 3.14159 * radius * radius;

       return area;
   }
   ```

4. We would like to define the `isEven` function to determine if a given integer n is even. Two function implementations are given below.

   (a)
   ```
   bool isEven(int n) {
       if (n%2 == 0) {
          return true;
       } else {
          return false;
       }
   }
   ```

   (b)
   ```
   bool isEven(int n) {
       if (n%2 == 0) {
          return true;
       }
       if (n%2 != 0) {
          return false;
       }
   }
   ```

   By writing a suitable `main` function, test if the above functions work. Rewrite the `isEven` function such that it makes use of only one return statement.

5. Credit card numbers are typically 16-digits long and must conform to the Luhn Algorithm which is also known as the modulus-10 algorithm. We illustrate the algorithm using the credit card number: 8765 4321 1357 2468

8	7	6	5	4	3	2	1	1	3	5	7	2	4	6	8
Double the values of every alternating digit starting from the second digit from the right.															
16	7	12	5	8	3	4	1	2	3	10	7	4	4	12	8
Add together the individual single digits of all the numbers obtained above.															
7	7	3	5	8	3	4	1	2	3	1	7	4	4	3	8
Sum up all the numbers:															
$7+7+3+5+8+3+4+1+$ $2+3+1+7+4+4+3+8=70$															

If the last digit of the Luhn sum is zero, then the credit card number is a valid one. Your task is to write a program that reads in a 16-digit credit card number as input from the user and outputs the last digit of the corresponding Luhn sum. The credit card number is to be read as four 4-digit numbers. *Why not just read a single 16-digit number?*.

(a) Notice that for each 4-digit number the method to compute the partial sum is the same. Write a function `partialSum` that takes in a 4-digit number `num` as argument and returns the partial sum of that 4-digit number.

```
int partialSum(int num);
```

(b) Write a function `luhnSum` that takes in a credit card number as four 4-digit numbers and computes the corresponding Luhn sum.

```
int luhnSum(int num1, int num2, int num3, int num4);
```

(c) Write a `main` function to read in the 16-digit credit card number and outputs the last digit of the corresponding Luhn sum. Sample runs are given below. User input is underlined.

- Sample run #1:

```
Enter credit card number: 8765 4321 1357 2468
The last digit of the Luhn sum is 0.
```

- Sample run #2:

```
Enter credit card number: 1234 5678 8765 4321
The last digit of the Luhn sum is 2.
```

6. (a) Write a function `getDistance` to compute and return the euclidean distance between two locations given by their Cartesian coordinates (x_1, y_1) and (x_2, y_2). Use the distance formula

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Declare suitable parameters to accept the two locations. Note that each location is given by a pair of real numbers.

(b) Everyday, Mr. Get A. Life will drive from home to office in the morning. After work, he makes a trip to the market for groceries before returning home. Suppose all three locations (home, office and market) are represented as Cartesian coordinates. Using the function developed in 6a, write a program to calculate the total distance traveled by Mr. Get A. Life.

A sample run is given as follows. User input is underlined.

```
Enter x and y coordinates for home: 1.5 2.0
Enter x and y coordinates for office: 4.25 5.75
Enter x and y coordinates for market: 7.0 10.20

The total distance is 19.755128
```