WEEK 3 - FUNCTIONS AND POINTERS

You are required to do the following:

1. **Lab Questions** – Please do the lab questions during the lab session. When doing your lab questions, please follow exactly the question requirements on program input/output as our automated assessment system is based on test cases using exact string matching on program input/output.

2. **Lab Assignment Questions** – Please do the assignment questions and submit your code to the online Automated Programming Assessment System (APAS) for grading.

Lab Tutor: For this lab-tutorial session, please discuss the solution for each question in the lab. You may allocate about 30 minutes for each question. No need to discuss the assignment questions.

Lab Questions

Questions 1-3

You may use the program template in Figure 1 to test your functions in the following three questions. The program contains a **main**() which includes a switch statement so that the following functions can be tested by the user. Write the code for each function and use the suggested test cases to test your code for correctness.

```
#include <stdio.h>
/* function prototypes */
int numDigits1(int num);
int digitPos1(int num, int digit);
int square1(int num);
void numDigits2(int num, int *result);
void digitPos2(int num, int digit, int *result);
void square2(int num, int *result);

int main()
{
   int choice;
   int number, digit, result=0;
   do {
      printf("\nPerform the following functions ITERATIVELY:\n");
      printf("1:  numDigits1()\n");
      printf("2:  numDigits2()\n");
      printf("3:  digitPos1()\n");
      printf("4:  digitPos2()\n");
      printf("5:  square1()\n");
      printf("6:  square2()\n");
      printf("7:  quit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("numDigits1(): %d\n", numDigits1(number));
            break;
         case 2:
            printf("Enter the number: \n");
            scanf("%d", &number);
            numDigits2(number, &result);
            printf("numDigits2(): %d\n", result);
```

```
            break;
         case 3:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("Enter the digit: \n");
            scanf("%d", &digit);
            printf("digitPos1(): %d\n", digitPos1(number, digit));
            break;
         case 4:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("Enter the digit: \n");
            scanf("%d", &digit);
            digitPos2(number, digit, &result);
            printf("digitPos2(): %d\n", result);
            break;
         case 5:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("square1(): %d\n", square1(number));
            break;
         case 6:
            printf("Enter the number: \n");
            scanf("%d", &number);
            square2(number, &result);
            printf("square2(): %d\n", result);
            break;
         default: printf("Program terminating .....\n");
            break;
      }
   } while (choice < 7);
   return 0;
}
/* add function code here */
   int numDigits1(int num)
   {
      int count = 0;

      do {
         count++;
         num = num/10;
      } while (num > 0);
      return count;
   }
   void numDigits2(int num, int *result)
   {
      *result=0;
      /* Write your program code here */
   }
   int digitPos1(int num, int digit)
   {
      /* Write your program code here */
   }
   void digitPos2(int num, int digit, int *result)
   {
      int pos=0;
      *result=0;
      do {
         pos++;
         if (num%10 == digit){
            *result = pos;
            break;
         }
         num = num/10;
```

```
    } while (num > 0);
}
int square1(int num)
{
   /* Write your program code here */
}
void square2(int num, int *result)
{
   /* Write your program code here */
}
```

Figure 1

1. **(numDigits)** Write a function that counts the number of digits for a non-negative integer. For example, 1234 has 4 digits. The function **numDigits1()** returns the result. The function prototype is given below:

```
int numDigits1(int num);
```

Write another function **numDigits2()** that passes the result through the pointer parameter, *result*. The function prototype is given below:

```
void numDigits2(int num, int *result);
```

For separate program testing: The following sample program template is given for testing the functions:

```
#include <stdio.h>
int numDigits1(int num);
void numDigits2(int num, int *result);
int main()
{
   int number, result=0;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("numDigits1(): %d\n", numDigits1(number));
   numDigits2(number, &result);
   printf("numDigits2(): %d\n", result);
   return 0;
}
int numDigits1(int num)
{
   /* Write your program code here */
}
void numDigits2(int num, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the number:
1
numDigits1(): 1
numDigits2(): 1
```

(2) Test Case 2:

```
Enter the number:
13579
numDigits1(): 5
numDigits2(): 5
```

2. **(digitPos)** Write the function **digitPos1()** that returns the position of the first appearance of a specified digit in a positive number. The position of the digit is counted from the right and starts from 1. If the required digit is not in the number, the function should return 0. For example, digitPos1(12315, 1) returns 2 and digitPos1(12, 3) returns 0. The function prototype is given below:

```
int digitPos1(int num, int digit);
```

Write another function **digitPos2()** that passes the result through the pointer parameter, *result*. For example, if num = 12315 and digit = 1, then *result = 2 and if num=12 and digit = 3, then *result = 0. The function prototype is given below:

```
void digitPos2(int num, int digit, int *result);
```

For separate program testing: The following sample program template is given for testing the functions:

```
#include <stdio.h>
int digitPos1(int num, int digit);
void digitPos2(int num, int digit, int *result);
int main()
{
   int number, digit, result=0;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("Enter the digit: \n");
   scanf("%d", &digit);
   printf("digitPos1(): %d\n", digitPos1(number, digit));
   digitPos2(number, digit, &result);
   printf("digitPos2(): %d\n", result);
   return 0;
}
int digitPos1(int num, int digit)
{
   /* Write your program code here */
}
void digitPos2(int num, int digit, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the number:
234567
Enter the digit:
6
digitPos1(): 2
digitPos2(): 2
```

(2) Test Case 2:

```
Enter the number:
234567
Enter the digit:
8
digitPos1(): 0
digitPos2(): 0
```

3. **(square)** Write a function **square1()** that returns the square of a positive integer number *num*, by computing the sum of odd integers starting with 1 as shown in the example below. The result is returned to the calling function. For example, if *num* = 4, then $4^2 = 1 + 3 + 5 + 7 = 16$ is returned; if *num* = 5, then $5^2 = 1 + 3 + 5 + 7 + 9 = 25$ is returned. The function prototype is:

```
int square1(int num);
```

Write another function **square2()** that passes the result through the pointer parameter, *result*. For example, if *num* = 4, then *result = $4^2 = 1 + 3 + 5 + 7 = 16$; if *num* = 5, then *result = $5^2 = 1 + 3 + 5 + 7 + 9 = 25$. The function prototype is:

```
void square2(int num, int *result);
```

For separate program testing: The following sample program template is given for testing the functions:

```
#include <stdio.h>
int square1(int num);
void square2(int num, int *result);
int main()
{
   int number, result=0;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("square1(): %d\n", square1(number));
   square2(number, &result);
   printf("square2(): %d\n", result);
   return 0;
}
int square1(int num)
{
   /* Write your program code here */
}
void square2(int num, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the number:
4
square1(): 16
square2(): 16
```

(2) Test Case 2:

```
Enter the number:
0
square1(): 0
square2(): 0
```

4. **(calDistance)** Write a C program that accepts four decimal values representing the coordinates of two points, i.e. (x1, y1) and (x2, y2), on a plane, and calculates and displays the distance between the points:

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

Your program should be implemented using functions. Provide two versions of the function for calculating the distance: (a) one uses call by value only for passing parameters; and (b) the other uses call by reference to pass the result to the calling function.

The function prototypes are given below:

```
void inputXY(double *x1, double *y1, double *x2, double *y2);
void outputResult(double dist);
double calDistance1(double x1, double y1, double x2, double y2);
void calDistance2(double x1, double y1, double x2, double y2,
                  double *dist);
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
#include <math.h>
void inputXY(double *x1, double *y1, double *x2, double *y2);
void outputResult(double dist);
double calDistance1(double x1, double y1, double x2, double y2);
void calDistance2(double x1, double y1, double x2, double y2, double *dist);
int main()
{
   double x1, y1, x2, y2, distance=-1;

   inputXY(&x1, &y1, &x2, &y2);                  // call by reference
   distance = calDistance1(x1, y1, x2, y2);      // call by value
   printf("calDistance1(): ");
   outputResult(distance);
   calDistance2(x1, y1, x2, y2, &distance);      // call by reference
   printf("calDistance2(): ");
   outputResult(distance);                       // call by value
   return 0;
}
void inputXY(double *x1, double *y1, double *x2, double *y2)
{
   /* Write your code here */
}
void outputResult(double dist)
{
   /* Write your code here */
}
double calDistance1(double x1, double y1, double x2, double y2)
{
   /* Write your code here */
}
void calDistance2(double x1, double y1, double x2, double y2, double
*dist)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Input x1 y1 x2 y2:
1 1 5 5
calDistance1(): 5.66
calDistance2(): 5.66
```

(2) Test Case 2:

```
Input x1 y1 x2 y2:
-1 -1 5 5
calDistance1(): 8.49
calDistance2(): 8.49
```