WEEK 3 - FUNCTIONS AND POINTERS

You are required to do the following:

1. Lab Questions – please do the lab questions during the lab session.
2. Assignment Questions – please do the assignment questions and submit your code to the online automated programming assignment submission system (APAS) for grading.

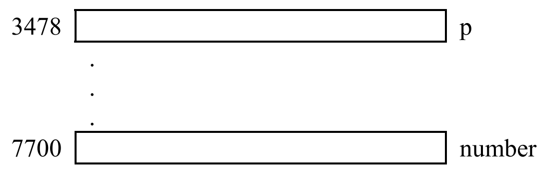
Lab Tutor: For this lab-tutorial session, please discuss about the solution for each question in the lab. You may allocate about 25 minutes for each question. No need to discuss about the assignment questions.

Lab Questions

1. Assume the following declaration:

int number;
int *p;

Assume also that the address of number is 7700 and the address of p is 3478. That is,



For each case below, determine the value of

(a) number (b) &number (c) p (d) &p (e) *p

All of the results are cumulative.

(i) p = 100; number = 8
(ii) number = p
(iii) p = &number
(iv) *p = 10
(v) number = &p
(vi) p = &p

Questions 2-4

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

2. **(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);
```

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
```

For separate program testing: The following sample program template is given below:

```
#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 */
}
```

3. **(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);
```

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
```

For separate program testing: The following sample program template is given below:

```
#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 */
}
```

4. **(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);
```

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
```

For separate program testing: The following sample program template is given below:

```
#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 */
}
```

5. What will be the output of the following program?

```
#include  <stdio.h>

void function0();
void function1(int h, int k);
void function2(int *h, int *k);

int main()
{
   int h, k;

   h = 5;
   k = 15;
   printf(“h = %d, k = %d\n”, h, k);      /* line (i) */
   function0();
   printf(“h = %d, k = %d\n”, h, k);      /* line (ii) */
   function1(h, k);
   printf(“h = %d, k = %d\n”, h, k);      /* line (iii) */
   function2(&h, &k);
   printf(“h = %d, k = %d\n”, h, k);      /* line (iv) */

   return 0;
}

void function0()
{
   int h, k;

   h = k = -100;
   printf(“h = %d, k = %d\n”, h, k);      /* line (v) */
}

void function1(int h, int k)
{
   printf(“h = %d, k = %d\n”, h, k);      /* line (vi) */
   h = k = 100;
   printf(“h = %d, k = %d\n”, h, k);      /* line (vii) */
}

void function2(int *h, int *k)
{
   printf(“h = %d, k = %d\n”, *h, *k);    /* line (viii) */
   *h = *k = 200;
   printf(“h = %d, k = %d\n”, *h, *k);    /* line (ix) */
}
```

Assignment Questions for Submission

You are required to submit your code for the following assignment questions to the Automated Programming Submission and Assessment System (APAS) for marking and grading. For example, if your lab session is held on the week of 14 August 2017 (any day of the week), then the deadline for your submission will be on Friday the following week 25 August 2017 (Friday), at 11.59pm.

1. **(digitValue)** Write a function that returns the value of the *k*th digit (k>0) from the right of a non-negative integer *num*. For example, if num is1234567 and k is 3, the function will return 5 and if num is 1234 and k is 8, the function will return 0. Write the function in two versions. The function **digitValue1()** returns the result, while **digitValue2()** passes the result through pointer parameter *result*. The prototypes of the function are given below:

```
int digitValue1(int num, int k);
void digitValue2(int num, int k, int *result);
```

Write a C program to test the functions.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the number:
234567
Enter k position:
3
digitValue1(): 5
digitValue2(): 5
```

(2) Test Case 2:
```
Enter the number:
123
Enter k position:
8
digitValue1(): 0
digitValue2(): 0
```

A sample program to test the function is given below:

```
#include <stdio.h>
int digitValue1(int num, int k);
void digitValue2(int num, int k, int *result);
int main()
{
   int num, digit, result=-1;

   printf("Enter the number: \n");
   scanf("%d", &num);
   printf("Enter k position: \n");
   scanf("%d", &digit);
   printf("digitValue1(): %d\n",  digitValue1(num, digit));
   digitValue2(num, digit, &result);
   printf("digitValue2(): %d\n", result);
   return 0;
}
int digitValue1(int num, int k)
{
   /* Write your program code here */
}
void digitValue2(int num, int k, int *result)
{
   /* Write your program code here */
}
```

2. **(power)** Write a function that computes the power of a number. The power may be any integer value. Write two iterative versions of the function. The function **power1()** returns the computed result, while **power2()** passes the result through the pointer parameter *result*. The function prototypes are given below:

```
float power1(float num, int p);
void power2(float num, int p, float *result);
```

Write a C program to test the functions.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the number and power:
2 3
power1(): 8.00
power2(): 8.00
```

(2) Test Case 2:
```
Enter the number and power:
2 -4
power1(): 0.06
power2(): 0.06
```

(3) Test Case 3:
```
Enter the number and power:
2 0
power1(): 1.00
power2(): 1.00
```

A sample program to test the function is given below:

```
#include <stdio.h>
float power1(float num, int p);
void power2(float num, int p, float *result);
int main()
{
   int power;
   float number, result=-1;

   printf("Enter the number and power: \n");
   scanf("%f %d", &number, &power);
   printf("power1(): %.2f\n", power1(number, power));
   power2(number,power,&result);
   printf("power2(): %.2f\n", result);
   return 0;
}
float power1(float num, int p)
{
   /* Write your program code here */
}
void power2(float num, int p, float *result)
{
   /* Write your program code here */
}
```

3. (**gcd – greatest common divisor**) Write a C function gcd() that computes the greatest common divisor. For example, if num1 is 4 and num2 is 7, then the function will return 1; if num1 is 4 and num2 is 32, then the function will return 4; and if num1 is 4 and num2 is 38, then the function will return 2. Write two iterative versions of the function. The function **gcd11()** returns the computed result, while **gcd2()** passes the result through the pointer parameter *result*. The function prototypes are given as follows:

```
int gcd1(int num1, int num2);
void gcd2(int num1, int num2, int *result);
```

Write a C program to test the function.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter 2 numbers:
4 7
gcd1(): 1
gcd2(): 1
```

(2) Test Case 2:
```
Enter 2 numbers:
4 32
gcd1(): 4
gcd2(): 4
```

(3) Test Case 3:
```
Enter 2 numbers:
4 38
gcd1(): 2
gcd2(): 2
```

A sample program to test the function is given below:

```
#include <stdio.h>
int gcd1(int num1, int num2);
void gcd2(int num1, int num2, int *result);
int main()
{
   int x,y,result=-1;
   printf("Enter 2 numbers: \n");
   scanf("%d %d", &x, &y);
   printf("gcd1(): %d\n", gcd1(x, y));
   gcd2(x,y,&result);
   printf("gcd2(): %d\n", result);
   return 0;
}
int gcd1(int num1, int num2)
{
   /* Write your program code here */
}
void gcd2(int num1, int num2, int *result)
{
   /* Write your program code here */
}
```

4. **(countOddDigits)** Write a C function to count the number of odd digits, i.e. digits with values 1,3,5,7,9 in a non-negative number. For example, if num is 1234567, then 4 will be returned. Write the function in two versions. The function countOddDigits1() returns the result to the caller, while countOddDigits2() passes the result through the pointer parameter *result*. The function prototypes are given below:

```
int countOddDigits1(int num);
void countOddDigits2(int num, int *result);
```

Write a C program to test the functions.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the number:
34567
countOddDigits1(): 3
```

```
countOddDigits2(): 3
```

(2) Test Case 2:

```
Enter the number:
2468
countOddDigits1(): 0
countOddDigits2(): 0
```

A sample program to test the function is given below:

```
#include <stdio.h>
int countOddDigits1(int num);
void countOddDigits2(int num, int *result);
int main()
{
   int number, result=-1;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("countOddDigits1(): %d\n", countOddDigits1(number));
   countOddDigits2(number, &result);
   printf("countOddDigits2(): %d\n", result);
   return 0;
}
int countOddDigits1(int num)
{
   /* Write your program code here */
}
void countOddDigits2(int num, int *result)
{
   /* Write your program code here */
}
```