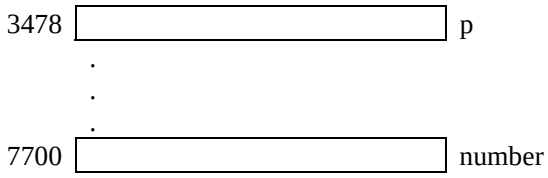
WEEK 3 TUTORIAL - FUNCTIONS AND POINTERS

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 **(i) (a) 8 (b) 7700 (c) 100 (d) 3478 (e) NULL**
(ii) number = p **(i) (a) 100 (b) 7700 (c) 100 (d) 3478 (e) NULL**
(iii) p = &number **(i) (a) 100 (b) 7700 (c) 7700 (d) 3478 (e) 100**
(iv) *p = 10 **(i) (a) 100 (b) 7700 (c) 7700 (d) 3478 (e) 10**
(v) number = &p **(i) (a) 3478 (b) 7700 (c) 7700 (d) 3478 (e) 10**
(vi) p = &p **(i) (a) 3478 (b) 7700 (c) 3478 (d) 3478 (e) 10**

2. 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;
```

h = 5, k = 15
h = -100, k = -100
h = 5, k = 15
h = 5, k = 15; h = 100, k = 100
h = 5, k = 15
h = 5, k = 15; h = 200, k = 200
h = 200, k = 200

```
   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) */
}
```

3. **(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);
```

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

```
#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 code here */
}
void digitValue2(int num, int k, int *result)
{
   /* Write your code here */
}
```

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:
234567
Enter k position:
1
digitValue1(): 7
digitValue2(): 7
```

(3) Test Case 3:

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