Week 3 Tutorial: Functions and Pointers – Suggested Solutions

Q1:

(i)

3478	100	p	p = 100
7700	8	number	number = 8

That is (a) number is 8 (b) &number is 7700 (c) p is 100 (d) &p is 3478 (e) *p is the content of the memory location 100.

(ii)

3478	100	p	
7700	100	number	number = p

That is (a) number is 100 (b) &number is 7700 (c) p is 100 (d) &p is 3478 (e) *p is the content of the memory location 100.

(iii)

3478	7700	p	p = &number
7700	100	number	

That is (a) number is 100 (b) &number is 7700 (c) p is 7700 (d) &p is 3478 (e) *p is 100.

(iv)

3478	7700	p	*p = 10
7700	10	number	

That is (a) number is 10 (b) &number is 7700 (c) p is 7700 (d) &p is 3478 (e) *p is 10.

(v)

3478	7700	p	
7700	3478	number	number = &p

That is (a) number is 3478 (b) &number is 7700 (c) p is 7700 (d) &p is 3478 (e) *p is 3478.

(vi)

3478	3478	p	p = &p
7700	3478	number	

That is (a) number is 3478 (b) &number is 7700 (c) p is 3478 (d) &p is 3478 (e) *p is 3478.

Q2:
The output:

	remark
h = 5, k = 15	line (i)
h = -100, k = -100	line (v)
h = 5, k = 15	line (ii)
h = 5, k = 15	line (vi)
h = 100, k = 100	line (vii)
h = 5, k = 15	line (iii)
h = 5, k = 15	line (viii)
h = 200, k = 200	line (ix)
h = 200, k = 200	line (iv)

Q3: (digitValue)

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

   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)
{
   int i, r;

   for (i=0; i<k; i++)
   {
      r = num%10;
      num /= 10;
   }
   return r;
}
void digitValue2(int num, int k, int *result)
{
   int i, r;

   for (i=0; i<k; i++)
   {
      r = num%10;
      num /= 10;
   }
   *result = r;
}
```

Q4: (calDistance)

```c
#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);
```

```c
int main()
{
   double x1, y1, x2, y2, distance;

   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)
{
   printf("Input x1 y1 x2 y2: \n");
   scanf("%lf %lf %lf %lf", x1, y1, x2, y2);
}
void outputResult(double dist)
{
   printf("%.2f\n", dist);
}
double calDistance1(double x1, double y1, double x2, double y2)
{
   x1 = x1 - x2;
   x1 = x1 * x1;
   y1 = y1 - y2;
   y1 = y1 * y1;
   return (sqrt(x1 + y1));
}
void calDistance2(double x1, double y1, double x2, double y2, double *dist)
{
   x1 = x1 - x2;
   x1 = x1 * x1;
   y1 = y1 - y2;
   y1 = y1 * y1;
   *dist = sqrt(x1 + y1);
}
```