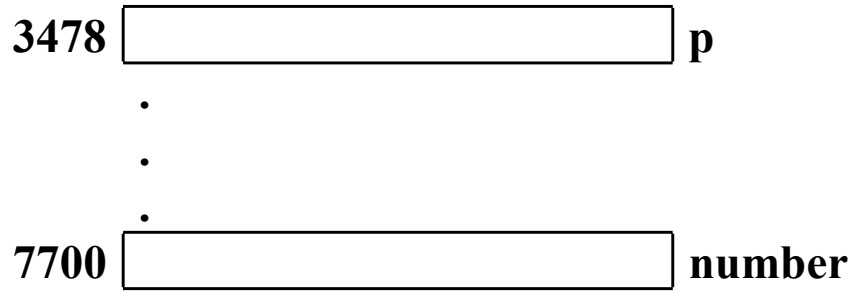
Tutorial 2
Functions and Pointers - Q1

Assume the following declaration:

```
int number;
int *p;
```

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



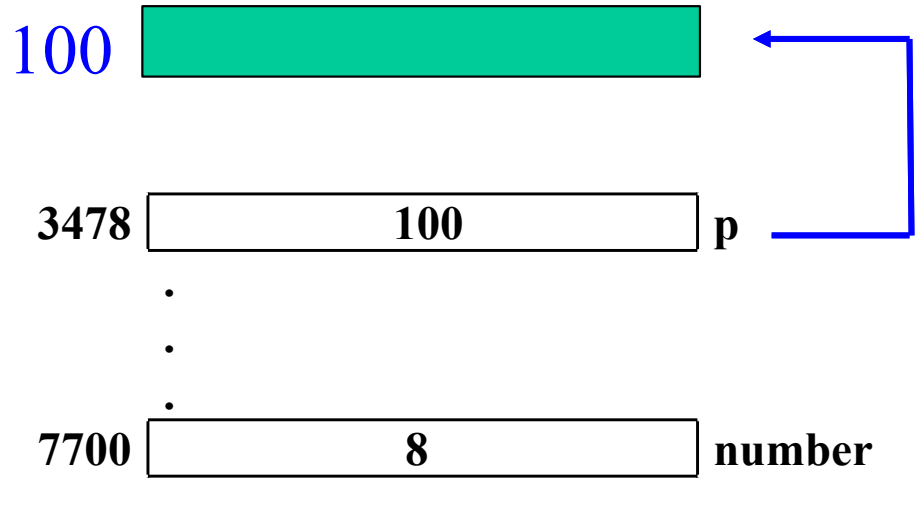
For each case below, determine the values of

(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

(i) p = 100; number = 8



p = 100

number = 8

That is,

(a) number is 8

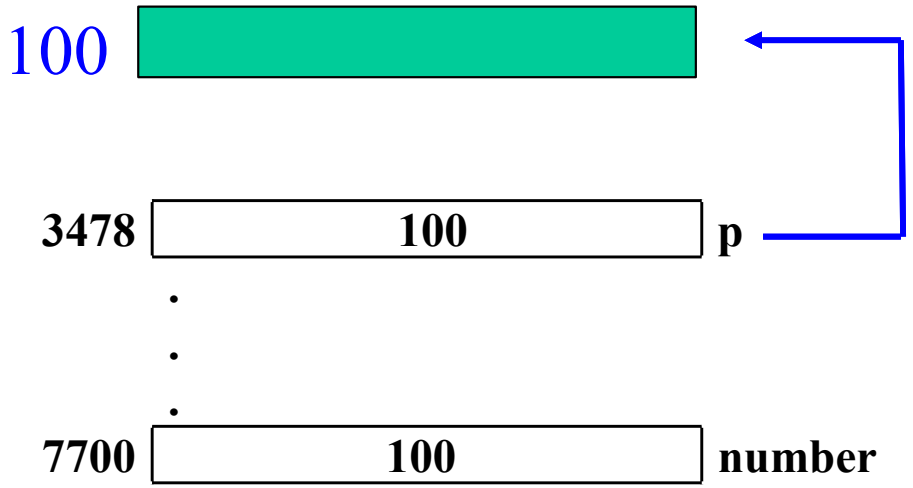
(b) &number is 7700

(c) p is 100

(d) &p is 3478

(e) *p is the content of the memory location 100.

(ii) number = p



number = p

That is,

(a) number is 100

(b) &number is 7700

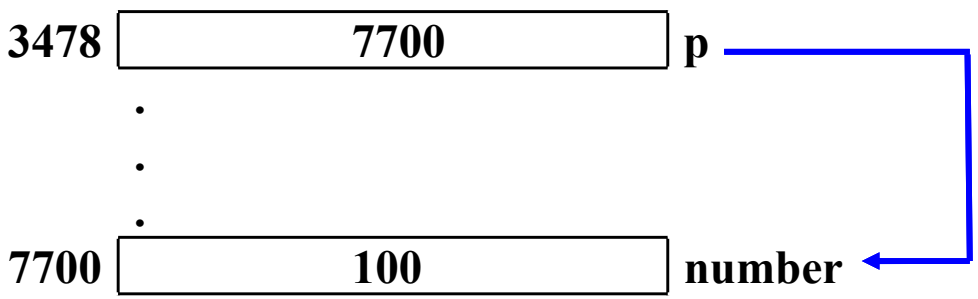
(c) p is 100

(d) &p is 3478

(e) *p is the content of the memory location 100

(iii) p = &number

p = &number



That is,

(a) number is 100

(b) &number is 7700

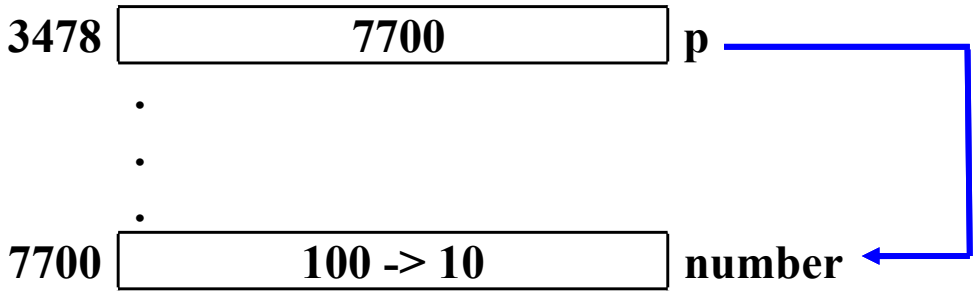
(c) p is 7700

(d) &p is 3478

(e) *p is 100

(iv) *p = 10

*p = 10



That is,

(a) number is 10

(b) &number is 7700

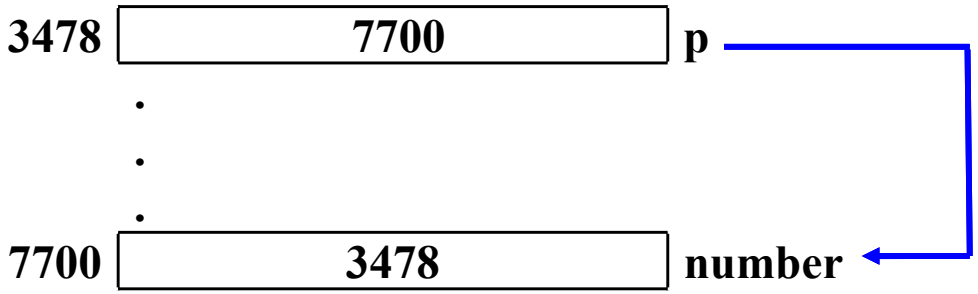
(c) p is 7700

(d) &p is 3478

(e) *p is 10

(v) number = &p

number = &p



That is,

(a) number is 3478

(b) &number is 7700

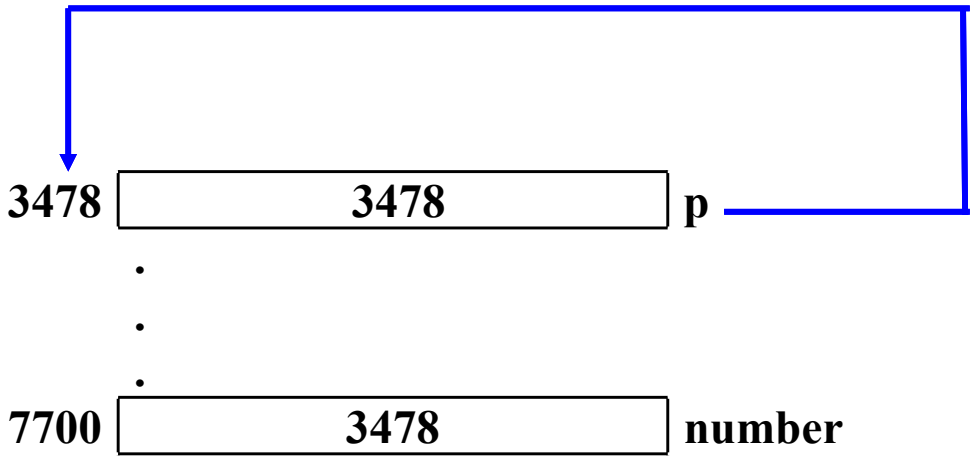
(c) p is 7700

(d) &p is 3478

(e) *p is 3478

(vi) p = &p

p = &p



That is,

(a) number is 3478

(b) &number is 7700

(c) p is 3478

(d) &p is 3478

(e) *p is 3478

Functions and Pointers - Q2

Find the error in each of the following program segments and explain how the error may be corrected.

Q2 – Suggested Answer

(a)

```
int product(int m, int n)
{
    int result;
    result =m * n;
}
```

Q2 – Suggested Answer

(a)

```
int product(int m, int n)
{
    int result;
    result =m * n;
}
```

Suggested Answer

error: result is **not returned** by the function.

correction: add the statement return result; as the last statement in the function.

(b)

```
int sumofSquare(int n)  /* assume n is non-negative */
{
      int sum = 0;
      if (n == 0)
            return 0;
      else
                  for (j = 1; j <= n; j++) sum += j * j ;
}
```

(b)

```
int sumofSquare(int n)  /* assume n is non-negative */
{
      int sum = 0;
      if (n == 0)
            return 0;
      else
                  for (j = 1; j <= n; j++) sum += j * j ;
}
```

Suggested Answer

error: when **n is not zero**, the function does not return the result. Also, **j is not declared**.

corrections: add in the declaration for j and the else part of the if statement is

```
            else {
                for (j = 1; j <= n; j++)          sum += j * j ;
                return sum;
            }
```

(c)

```
void  ft(float a)
{
      float a;
      printf(“%f\n”, a);
}
```

(c)

```
void  ft(float a)
{
      float a;
      printf("%f\n", a);
}
```

Suggested Answer

error: formal argument a is re-declared as the local variable a.

correction: change the name of the local variable a to something else.

(d)

```
void  height(float * h)
{
     scanf(“%f”, &h);
}
```

(d)

```
void  height(float * h)
{
     scanf("%f", &h);
}
```

Suggested Answer

error: the parameter **h** contains the address of the actual parameter, in other words, the value of h is the address of the actual parameter. This address should be passed to scanf( ) and not the address of h.

correction: remove the & in front of h.

(e)

```
void  height(float * h)
{
      scanf(“%f”, h);
      return *h;
}
```

(e)

```
void  height(float * h)
{
      scanf("%f", h);
      return *h;
}
```

Suggested Answer

error: the function is of type **void**. It should not return any value using the return statement.

correction: remove the return statement.

(f)

```
int divideBy4(int n)
{
    int divideBy2(int m)
    {
        return m/2;
    }
    return (divideBy2(divideBy2(n));
}
```

(f)

```
int divideBy4(int n)
{
    int divideBy2(int m)
    {
        return m/2;
    }
    return (divideBy2(divideBy2(n));
}
```

Suggested Answer

error: it is **not allowed** to define a function inside another function.

Correction: the definition for divideBy2( ) should be taken out of the function divideBy4( ).

Functions and Pointers - Q3

```
#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) */
}
```

What will be the output of the program?

Q3 – Suggested Answer

```
#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) */
}
```

(1) h = 5, k = 15 line (i)

```
#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) */
}
```

(1) h = 5, k = 15 line (i)

(2) h = -100, k = -100 line (v)

```
#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) */
}
```

(1) h = 5, k = 15 line (i)

(3) h = 5, k = 15 line (ii)

(2) h = -100, k = -100 line (v)

```
#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) */
}
```

(1) h = 5, k = 15 line (i)

(3) h = 5, k = 15 line (ii)

(2) h = -100, k = -100 line (v)

(4) h = 5, k = 15 line (vi)

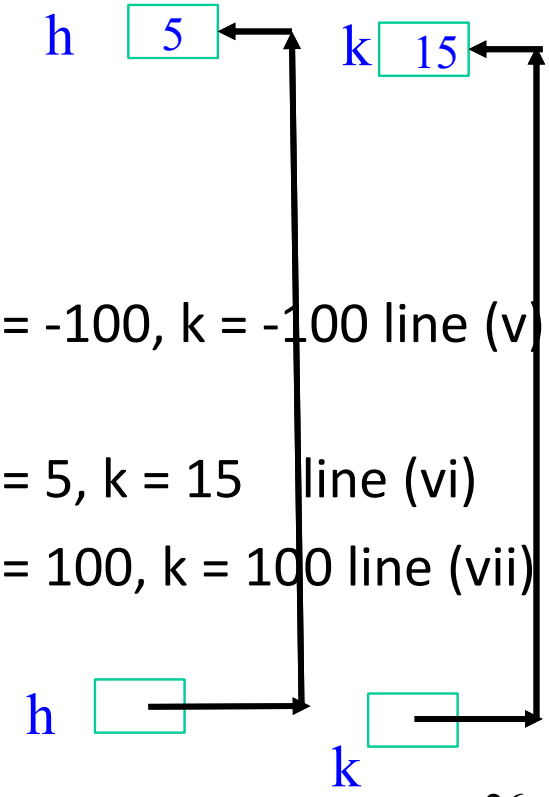
(5) h = 100, k = 100 line (vii)

```c
#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) */
}
```

(1) h = 5, k = 15 line (i)

(3) h = 5, k = 15 line (ii)

(6) h = 5, k = 15 line (iii)



(2) h = -100, k = -100 line (v)

(4) h = 5, k = 15 line (vi)

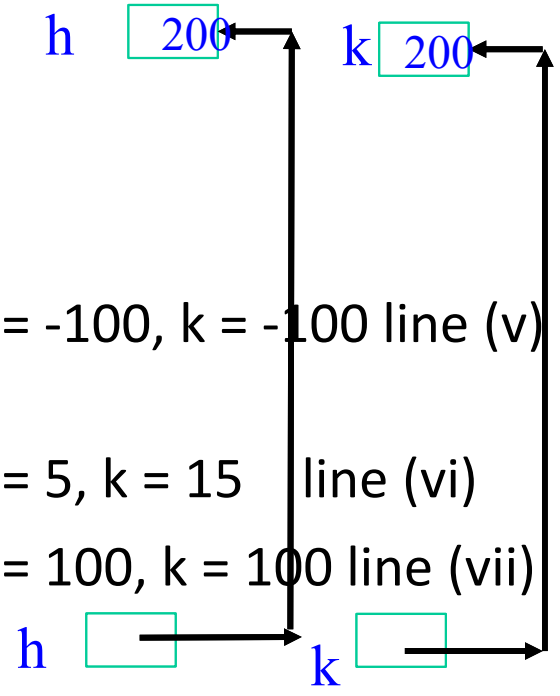
(5) h = 100, k = 100 line (vii)

```
#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) */
}
```

(1) h = 5, k = 15 line (i)

(3) h = 5, k = 15 line (ii)

(6) h = 5, k = 15 line (iii)



(2) h = -100, k = -100 line (v)

(4) h = 5, k = 15 line (vi)

(5) h = 100, k = 100 line (vii)

(7) h = 5, k = 15 line (viii)

```
#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) */
}
```

(1) h = 5, k = 15 line (i)

(3) h = 5, k = 15 line (ii)

(6) h = 5, k = 15 line (iii)

(9) h = 200, k = 200 line (iv)

(2) h = -100, k = -100 line (v)

(4) h = 5, k = 15 line (vi)

(5) h = 100, k = 100 line (vii)

(7) h = 5, k = 15 line (viii)

(8) h = 200, k = 200 line (ix)

```c
#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) */
}
```

The output:

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)

Functions and Pointers - Q4

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:

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 back.

A sample input and output session is given below:

```
Input x1 y1 x2 y2: 1 1 5 5
calDistance1()
Distance: 5.656854
calDistance2()
Distance: 5.656854
```

Q4 – Suggested Answer

```
#include <stdio.h>
#include <math.h>

void  inputXY(double *, double *, double *, double *);
double   calDistance1(double, double, double, double);
void   calDistance2(double, double, double, double, double*);
void  outputResult(double);

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()\n");
    outputResult(distance);    // call by value

    calDistance2(x1, y1, x2, y2, &distance);   // call by reference
    printf("calDistance2()\n");
    outputResult(distance);     // call by value

    return 0;
}
```

Using Call by Reference

```
void inputXY(double *x1, double *y1, double *x2, double *y2)
{
    printf("Input x1 y1 x2 y2: ");
    scanf("%lf %lf %lf %lf", x1, y1, x2, y2);
 }
```

/* with call by reference, the function inputXY() will be able to pass the values of 4 variables to the calling function */

User Input:

```
Input x1, y1, x2, y2: 5 10 15 20
```

Note: more than 1 input to be returned

inputXY – you may return more than one value to the calling function via the pointer variables

Using Call by Value

```
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));
 }
```

Using Call by Reference

```
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);
 }
```

```
void outputResult(double dist2)
{
    printf("Distance: %f\n", dist2);
 }
```