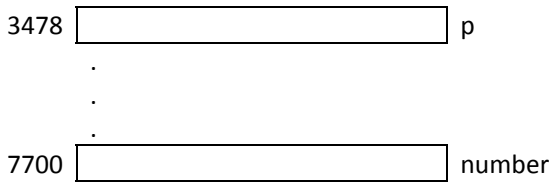
Tutorial 2 – Functions and Pointers – Suggested Answers

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

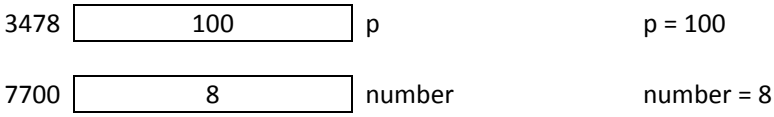
(a) number (b) &number (c) p (d) &p (e) *p

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

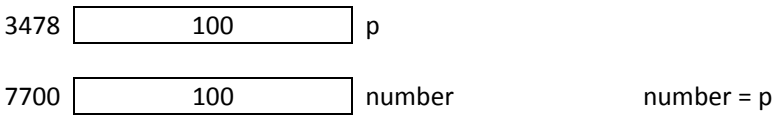
Suggested Answer:

(i)



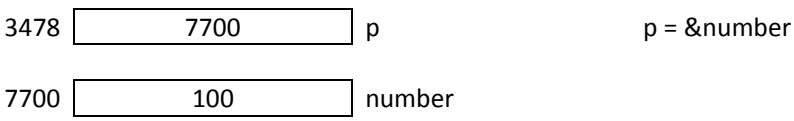
That is (a) number is 8 (b) &number is 7700 (c) p is 100 (d) &p is 3478 (e) *p is the content of the memory location 100.

(ii)



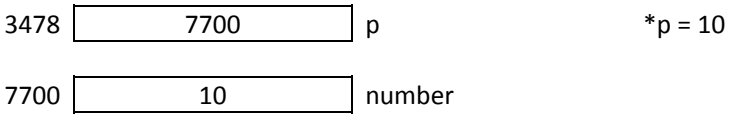
That is (a) number is 100 (b) &number is 7700 (c) p is 100 (d) &p is 3478 (e) *p is the content of the memory location 100.

(iii)



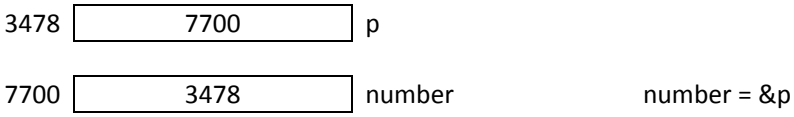
That is (a) number is 100 (b) &number is 7700 (c) p is 7700 (d) &p is 3478 (e) *p is 100.

(iv)



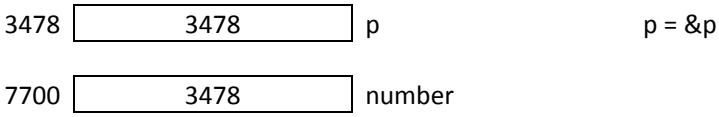
That is (a) number is 10 (b) &number is 7700 (c) p is 7700 (d) &p is 3478 (e) *p is 10.

(v)



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

(vi)



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

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

(a)
```
int product(int m, int n)
{
int result;

        result =m * n;
}
```

Suggested Answer: (a) 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 ;
}
```

Suggested Answer: (b) 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);
}
```

Suggested Answer: (c) 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);
}
```

Suggested Answer: (d) 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;
}
```

Suggested Answer: (e) 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));
 }
```

Suggested Answer: (f) 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( ).

3. 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) */
}
```

Suggested Answer:

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)

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.

Suggested Answer:

```
#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;

  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);
}
```