# Tutorial 5A- Solution

**Objectives:** To practice with Arrays

## Write C statements to do the following:

* 1. Declare an array myArray of 10 elements of the type int.

**int myArray[10];**

* 1. Output the value of the fifth element of the array myArray using printf() function.

**printf(“%d”, myArray[4] );**

* 1. Set the value of the third element of an array myArray to 27.

**myArray[2] = 27;**

* 1. Set the value of an array's eighth element equal to the sum of the second and the seventh elements

## What is the output of the following code (when embedded in a complete and correct program)?

**myArray[7] = myArray[1] + myArray[6];**

int i, temp[10];

for (i= 0; i<10; i++) temp[i]=2\*i;

i = 0 1 2 3 4 .. . .

temp[i] = 0 2 4 6 8 . . .

for(i=0; i<10; i++)

printf( "%d ", temp[i] );

Output

**0 2 4 6 . . . 18**

:

**3. Given the declaration**

int zipCode[50];

int j;

char name[] = "Paul";

## Indicate attempts to access array elements beyond the bounds.

a. for(j = 0; j <= 49; j++)

zipCode[j] = 0;

b. for(j = 50; j >= 0; j--)

zipCode[j-1]= 0;

c. for(j = 0; j <= 50; j++)

zipCode[j] = 0;

1. Correct
2. Wrong as the last index will be -1
3. Wrong as the last index will be 50

## Write C code that compares two arrays.

#define SIZE 5

int data1[SIZE] = { 1, 2, 3, 4, 5 };

int data2[SIZE] = { 1, 2, 3, 4, 0 };

for( i=0; i< SIZE; i++ )

**// if a pair of elements are different we exit the loop**

if( data1[i] != data2[i] )

break;

if( i== SIZE )

**// WE LOOPED TO THE LAST ELEMENT OF THE LOOP**

**printf**("equal\n");

else

**printf**("NOT Equal\n");

Another solution:

**for**( i=0; i<SIZE && data1[i]==data2[i]; i++ );

if( i < SIZE ) **printf**("Notequal\n");

else

**printf**("Equal\n");

The above solution uses the empty statement “;”. The loop keeps running with an empty body as long as the condition:

“i<SIZE && data1[i]==data2[i]” remains true

Test Program:

**int** unequal=0; // Boolean flag

**int** data1[SIZE] = { 1, 2, 3, 4, 5 };

**int** data2[SIZE] = { 1, 2, 3, 4, 5 };

**for** (int i=0; i<SIZE;i++)

**if** (data1[i]!=data2[i])

{

unequal=1;

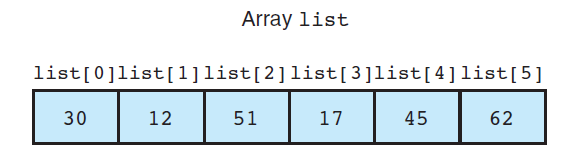
**break;**

}

**if**(unequal) **printf**("arrays are unequal\n");

**else** printf("arrays are equal");

1. **Write a for loop that sums the odd values from the LIST\_SIZE element array list. For example, the sum for this list would be 113 (51 + 17 + 45).**



**Answer:**

**int** i, sum;

sum = 0;

**for** (i = 0; i < LIST\_SIZE; ++i)

**if** (list[i] % 2 == 1)

sum += list[i];

1. **What is the difference in the use of array b that is implied by these two prototypes?**

**int fun\_one(int b[], n) ;**

**int fun\_two(const int b[], n) ;**

**Answer:**

In fun\_one, b can be used as an output parameter or as an input/output parameter. In fun\_two, b is strictly an input parameter array which can’t be amended within the function unlike with fun\_one function

1. **Define a function multiply that computes and returns the product of the type int elements of its array input argument. The function should have a second input argument telling the number of array elements to use.**

**Answer:**

/\*

\* Computes product of first size elements of array a.

\*/

**int** multiply(**const int** a[], **int** size)

{

**int** product, i;

product = 1;

**for** (i = 0; i < size; ++i)

product \*= a[i];

**return** (product);

}

1. **Write a program segment to display the sum of the values in each row of a 5 × 3 type double array named table. How many row sums will be displayed? How many elements are included in each sum?**

for (t = 0; t < 5; ++t) {

sum = 0.0;

for (t1 = 0; t1 < 3; ++t1)

sum += table[t][t1];

printf("Row %d sum is %.2f.\n", t + 1, sum);

}

Five row sums are displayed.

Each sum includes three elements.

1. **Write a function that displays the values on the diagonal of its 10 × 10 matrix parameter.**

/\*

\* Displays the values on the diagonal of matrix a

\*/

void

print\_diag(int a[10][10])

{

int i;

for (i = 0; i < 10; ++i)

printf("Element(%d,%d): %d\n", i, i, a[i][i]);

}