# Week 4 Lab Tutorial: Arrays – Suggested Solutions

## Lab Questions

**Q1: (reverseAr1D)**

```
#include <stdio.h>
void printReverse1(int ar[], int size);
void printReverse2(int ar[], int size);
void reverseAr1D(int ar[], int size);
int main()
{
   int ar[10];
   int size, i;

   printf("Enter array size: \n");
   scanf("%d", &size);
   printf("Enter %d array: \n", size);
   for (i=0; i <= size-1; i++)
      scanf("%d", &ar[i]);
   printReverse1(ar, size);
   printReverse2(ar, size);
   reverseAr1D(ar, size);
   printf("reverseAr1D(): ");
   if (size > 0) {
      for (i=0; i<size; i++)
         printf("%d ", ar[i]);
   }
   return 0;
}
void printReverse1(int ar[], int size)
{
   int i;
   printf("printReverse1(): ");
   if (size > 0) {
      for (i=size-1; i>=0; i--)
         printf("%d ", ar[i]);
   }
   printf("\n");
}
void printReverse2(int ar[], int size)
{
   int i;
   printf("printReverse2(): ");
   if (size > 0) {
      for (i=size-1; i>=0; i--)
         printf("%d ", *(ar+i));
   }
   printf("\n");
}
/* reverseAr reverses the array contents and passes that back to the
calling function */
void reverseAr1D(int ar[], int size)
{
```

```
   int i, temp;
   if (size > 0) {
      for (i=0; i<size/2; i++){
         temp = ar[i];
         ar[i] = ar[size-i-1];
         ar[size-i-1] = temp;
      }
   }
}
```

**Q2: (swap2RowsCols2D)**

```
#include <stdio.h>
#define SIZE 3
void swap2Rows(int ar[][SIZE], int r1, int r2);
void swap2Cols(int ar[][SIZE], int c1, int c2);
void display(int ar[][SIZE]);
int main()
{
   int array[SIZE][SIZE];
   int row1, row2, col1, col2;
   int i,j;

   printf("Enter the matrix (3x3): \n");
   for (i=0; i<SIZE; i++)
      for (j=0; j<SIZE; j++)
         scanf("%d", &array[i][j]);
   printf("Enter two rows for swapping: \n");
   scanf("%d %d", &row1, &row2);
   swap2Rows(array, row1, row2);
   printf("The new array is: \n");
   display(array);
   printf("Enter two columns for swapping: \n");
   scanf("%d %d", &col1, &col2);
   swap2Cols(array, col1, col2);
   printf("The new array is: \n");
   display(array);
   return 0;
}
void display(int ar[][SIZE])
{
   int l,m;
   for (l = 0; l < SIZE; l++) {
      for (m = 0; m < SIZE; m++)
         printf("%d ", ar[l][m]);
      printf("\n");
   }
}
void swap2Rows(int ar[][SIZE], int r1, int r2)
/* swaps row ar[r1] with row ar[r2] */
{
   int temp;
   int n;

   for(n = 0; n < SIZE; n++) {
      temp = ar[r1][n] ;
      ar[r1][n] = ar[r2][n];
      ar[r2][n] = temp;
   }
}
```

```
void swap2Cols(int ar[][SIZE], int c1, int c2)
/* swaps column ar[][c1] with column ar[][c2] */
{
   int temp;
   int n;

   for(n = 0; n < SIZE; n++) {
      temp = ar[n][c1] ;
      ar[n][c1] = ar[n][c2];
      ar[n][c2] = temp;
   }
}
```

**Q3: (reduceMatrix2D)**

```
#include <stdio.h>
#define SIZE 10
void reduceMatrix2D(int ar[][SIZE], int rowSize, int colSize);
void display(int ar[][SIZE], int rowSize, int colSize);
int main()
{
   int ar[SIZE][SIZE], rowSize, colSize;
   int i,j;
   printf("Enter row size of the 2D array: \n");
   scanf("%d", &rowSize);
   printf("Enter column size of the 2D array: \n");
   scanf("%d", &colSize);
   printf("Enter the matrix (%dx%d): \n", rowSize, colSize);
   for (i=0; i<rowSize; i++)
      for (j=0; j<colSize; j++)
         scanf("%d", &ar[i][j]);

   reduceMatrix2D(ar, rowSize, colSize);
   printf("reduceMatrix2D(): \n");
   display(ar, rowSize, colSize);
   return 0;
}
void display(int ar[][SIZE], int rowSize, int colSize)
{
   int l,m;
   for (l = 0; l < rowSize; l++) {
      for (m = 0; m < colSize; m++)
         printf("%d ", ar[l][m]);
      printf("\n");
   }
}
void reduceMatrix2D(int ar[][SIZE], int rowSize, int colSize)
{
   int i, j, sum; // i for row, j for column
   /* for each column */
   for (j = 0; j < colSize; j++){
      sum = 0;
      // process the row below matrix[j][j] of the column
      for (i = j+1; i < rowSize; i++){
         sum += ar[i][j];
         ar[i][j] = 0;
      }
      ar[j][j] += sum;
   }
}
```

**Q4:**

The function add1() have two parameters. The first one is an array address and the second one is the size of the array. So the function adds 1 to every element of the one dimensional array. When the function is called in the for statement at line a by

```
add1(array[h], 4);
```

array[h] is an one dimensional array of 4 integers. It is the (h+1)th row of the two dimensional array 'array'. In fact, array[h] is the address of the first element of the (h+1)th row. So every function call works on one row of the two dimensional array.

When the for statement at line a is replace by add1(array[0], 3*4), it is passing the address of the first element of the first row to add1() and telling the function that the array size is 12. So add1() works on an one dimensional array starting at array[0] and with 12 elements.