Lab 3 - Arrays

Lab session – One hour is scheduled for the lab session. There are 2 questions for this lab. In addition, there is also 1 practice question for you to try if you have extra time in the lab.

Note: You do not need to submit your code for this lab.

Lab Questions

1. (**findAr1D**) Write a function **findAr1D()** that returns the subscript of the **first appearance** of a target number in an array. For example, if ar = **{ 3,6,9,4,7,8 }**, then **findAr1D(6,ar,3)** will return 0 where 6 is the size of the array and 3 is the number to be found, and **findAr1D(6,ar,9)** will return 2. If the required number is not in the array, the function will return -1. The function prototype is given as follows:

   ```
   int findAr1D(int size, int ar[ ], int target);
   ```

   A sample program template is given below to test the function:

   ```c
   #include <stdio.h>
   #define INIT_VALUE -1000
   int findAr1D(int size, int ar[], int target);
   int main()
   {
     int ar[20];
     int size, i, target, result = INIT_VALUE;

     printf("Enter array size: \n");
     scanf("%d", &size);
     printf("Enter %d data: \n", size);
     for (i=0; i<=size-1; i++)
       scanf("%d", &ar[i]);
     printf("Enter the target number: \n");
     scanf("%d", &target);
     result = findAr1D(size, ar, target);
     if (result == -1)
       printf("findAr1D(): Not found\n");
     else
       printf("findAr1D(): %d", result);
     return 0;
   }
   int findAr1D(int size, int ar[], int target)
   {
     /* Write your code here */
   }
   ```

   Some sample input and output sessions are given below:

   (1) Test Case 1:
       Enter array size:

5
Enter 5 data:
1 2 3 4 5
Enter the target number:
3
findAr1D(): 2

(2) Test Case 2:
Enter array size:
1
Enter 1 data:
5
Enter the target number:
5
findAr1D(): 0

(3) Test Case 3:
Enter array size:
7
Enter 7 data:
1 3 5 7 9 11 15
Enter the target number:
15
findAr1D(): 6

(4) Test Case 4:
Enter array size:
7
Enter 7 data:
1 3 5 7 9 11 15
Enter the target number:
2
findAr1D(): Not found

2. **(swap2RowsCols2D)** Write the code for the following matrix functions:

```
void swap2Rows(int ar[][SIZE], int r1, int r2);
/* the function swaps the row r1 with the row r2 of a 2-dimensional array ar */

void swap2Cols(int ar[][SIZE], int c1, int c2);
/* the function swaps the column c1 with the column c2 of a 2-dimensional array ar
*/
```

You may assume that the input matrix is a 3x3 matrix, i.e. SIZE = 3.

A sample program template is given below to test the functions:

```
#include <stdio.h>
#define SIZE 3
void swap2Rows(int ar[][SIZE], int r1, int r2);
void swap2Cols(int ar[][SIZE], int c1, int c2);
```

```c
void display(int ar[][SIZE]);
int main()
{
  int array[SIZE][SIZE];
  int row1, row2, col1, col2;
  int i,j;
  int choice;

  printf("Select one of the following options: \n");
  printf("1: getInput()\n");
  printf("2: swap2Rows()\n");
  printf("3: swap2Cols()\n");
  printf("4: display()\n");
  printf("5: exit()\n");
  do {
    printf("Enter your choice: \n");
    scanf("%d", &choice);
    switch (choice) {
      case 1:
        printf("Enter the matrix (3x3): \n");
        for (i=0; i<SIZE; i++)
          for (j=0; j<SIZE; j++)
            scanf("%d", &array[i][j]);
        break;
      case 2:
        printf("Enter two rows for swapping: \n");
        scanf("%d %d", &row1, &row2);
        swap2Rows(array, row1, row2);
        printf("The new array is: \n");
        display(array);
        break;
      case 3:
        printf("Enter two columns for swapping: \n");
        scanf("%d %d", &col1, &col2);
        swap2Cols(array, col1, col2);
        printf("The new array is: \n");
        display(array);
        break;
      case 4:
        display(array);
        break;
    }
  } while (choice < 5);
  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]);
```

```c
        printf("\n");
    }
}
void swap2Rows(int ar[][SIZE], int r1, int r2)
{
    /* Write your code here */
}
void swap2Cols(int ar[][SIZE], int c1, int c2)
{
    /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
Select one of the following options:
1: getInput()
2: swap2Rows()
3: swap2Cols()
4: display()
5: exit()
Enter your choice:
1
Enter the matrix (3x3):
5 10 15
15 20 25
25 30 35
Enter your choice:
2
Enter two rows for swapping:
1 2
The new array is:
5 10 15
25 30 35
15 20 25
Enter your choice:
5

(2) Test Case 2:
Select one of the following options:
1: getInput()
2: swap2Rows()
3: swap2Cols()
4: display()
5: exit()
Enter your choice:
1
Enter the matrix (3x3):
5 10 15
15 20 25
25 30 35

Enter your choice:

3

Enter two columns for swapping:

1 2

The new array is:

5 15 10

15 25 20

25 35 30

Enter your choice:

5

(3) Test Case 3:

Select one of the following options:

1: getInput()

2: swap2Rows()

3: swap2Cols()

4: display()

5: exit()

Enter your choice:

1

Enter the matrix (3x3):

1 2 3

4 5 6

7 8 9

Enter your choice:

2

Enter two rows for swapping:

0 2

The new array is:

7 8 9

4 5 6

1 2 3

Enter your choice:

3

Enter two columns for swapping:

0 2

The new array is:

9 8 7

6 5 4

3 2 1

Enter your choice:

5

(4) Test Case 4:

Select one of the following options:

1: getInput()

2: swap2Rows()

3: swap2Cols()

4: display()

5: exit()

```
Enter your choice:
1
Enter the matrix (3x3):
1 2 3
4 5 6
7 8 9
Enter your choice:
2
Enter two rows for swapping:
1 2
The new array is:
1 2 3
7 8 9
4 5 6
Enter your choice:
3
Enter two columns for swapping:
1 2
The new array is:
1 3 2
7 9 8
4 6 5
Enter your choice:
5
```

Practice Questions

3. **(reverseAr1D)** Write a C function printReverse() that prints an array of integers in reverse order. For example, if ar[5] = {1,2,3,4,5}, then the output 5,4,3,2,1 will be printed after applying the function printReverse(). The function prototype is given as follows:

    void printReverse(int ar[], int size);

    where size indicates the size of the array.

    Write two versions of printReverse(). One version printReverse1() uses the index notation and the other version printReverse2() uses the pointer notation for accessing the element of each index location.

    In addition, Write a C function reverseAr1D() that takes in an array of integers **ar** and an integer *size* as parameters. The parameter *size* indicates the size of the array to be processed. The function converts the content in the array in reverse order and passes the array to the calling function via call by reference.

    void reverseAr1D(int ar[ ], int size);

    A sample program template is given below for testing the functions:

    ```c
    #include <stdio.h>
    void printReverse1(int ar[], int size);
    void printReverse2(int ar[], int size);
    ```

```c
void reverseAr1D(int ar[], int size);
int main()
{
    int ar[80];
    int size, i;

    printf("Enter array size: \n");
    scanf("%d", &size);
    printf("Enter %d data: \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)
{
    /* Write your code here – using index */
}
void printReverse2(int ar[], int size)
{
    /* Write your code here – using pointer */
}
void reverseAr1D(int ar[ ], int size)
{
    /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
Enter array size:
5
Enter 5 data:
1 2 3 6 7
printReverse1(): 7 6 3 2 1
printReverse2(): 7 6 3 2 1
reverseAr1D(): 7 6 3 2 1

(2) Test Case 2:
Enter array size:
1
Enter 1 data:
5
printReverse1(): 5

printReverse2(): 5
reverseAr1D(): 5

(3) Test Case 3:
Enter array size:
7
Enter 7 data:
1 2 3 4 5 6 7
printReverse1(): 7 6 5 4 3 2 1
printReverse2(): 7 6 5 4 3 2 1
reverseAr1D(): 7 6 5 4 3 2 1

(4) Test Case 4:
Enter array size:
2
Enter 2 data:
2 4
printReverse1(): 4 2
printReverse2(): 4 2
reverseAr1D(): 4 2