# Arrays – Q1

(**findAr1D**) Write a function that returns the subscript of the first appearance of a target number in an array. For example, if array = **{3,6,9,4,7,8}**, then **findAr1D(6,array,3)** will return 0 where 6 is the size of the array and 3 is the number to be found, and **findAr1D(6,array,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:

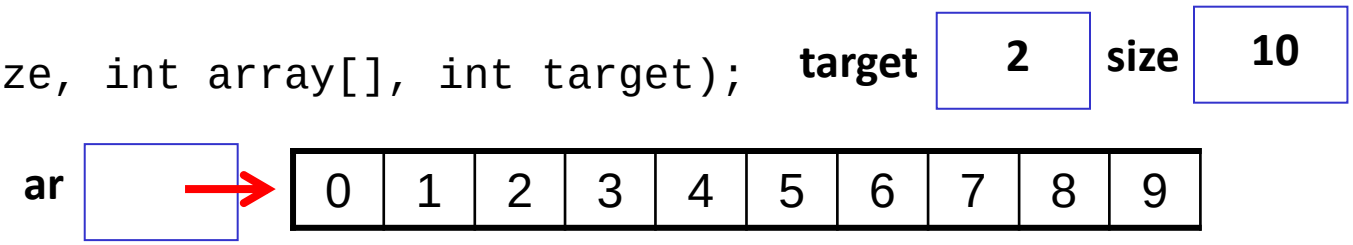
**int findAr1D(int size, int array[ ], int target);**

**Sample input and output sessions:**

```
(1) Test Case 1
Enter array size:
5
Enter 5 array:
1 2 3 4 5
Enter the target number:
2
findAr1D(): 1

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

# Arrays – Q1

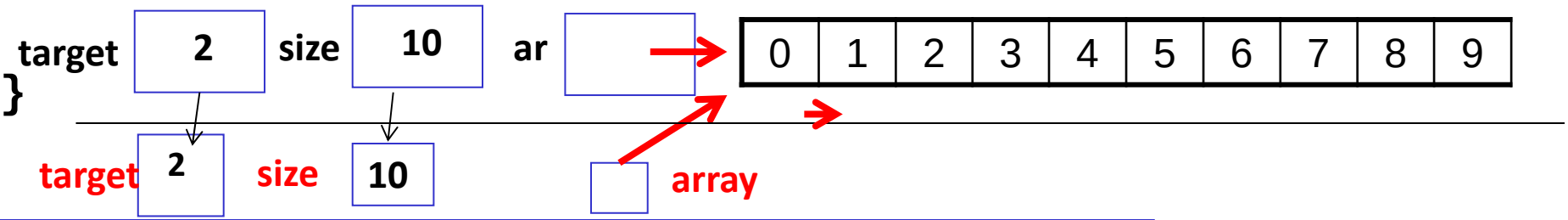
```
#include <stdio.h>
int findAr1D(int size, int array[], int target);
int main()
{
   int ar[20];
   int size, i, target, result=-1000;

   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]);
   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;
}
```




```
Enter array size:
10
Enter 5 array:
0 1 2 3 4 5 6 7 8 9
Enter the target number:
2
findAr1D(): 2
```

# Arrays – Q1

```
int main(){
}
```




```
int findAr1D(int size, int array[], int target)
{
   int j;
   for (j = 0; j < size; j++)
      if (array[j] == target)
         return j;
   return -1;
}
```

Using index

```
int findAr1D(int size, int array[], int target)
{
   int j;
   for (j = 0; j < size; j++)
      if (*(array+j) == target)
         return j;
   return -1;
}
```

Using pointer

# Arrays – Q2

**(swap2RowsCols2D)** Write the code for the following functions:

**void swap2Rows(int M[SIZE][SIZE], int r1, int r2);**

/* the function swaps the row r1 with the row r2 */

**void swap2Cols(int M[SIZE][SIZE], int c1, int c2);**

/* the function swaps the column c1 with the column c2 */

Write a C program to test the above functions. In addition, your program should print the resultant matrix after each operation. You may assume that the input matrix is a 3x3 matrix when testing the functions.

**Sample input and output :**

Enter the matrix (3x3):
***5 10 15***
***15 20 25***
***25 30 35***

**Enter two rows for swapping:**

***1 2***

The new array is:
5 10 15
25 30 35
15 20 25

**Enter two columns for swapping:**

***1 2***

The new array is:
5 15 10
25 35 30
15 25 20

# Arrays – Q2

```
#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;
   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;
```

Enter the matrix (3x3):

***5 10 15***

***15 20 25***

***25 30 35***

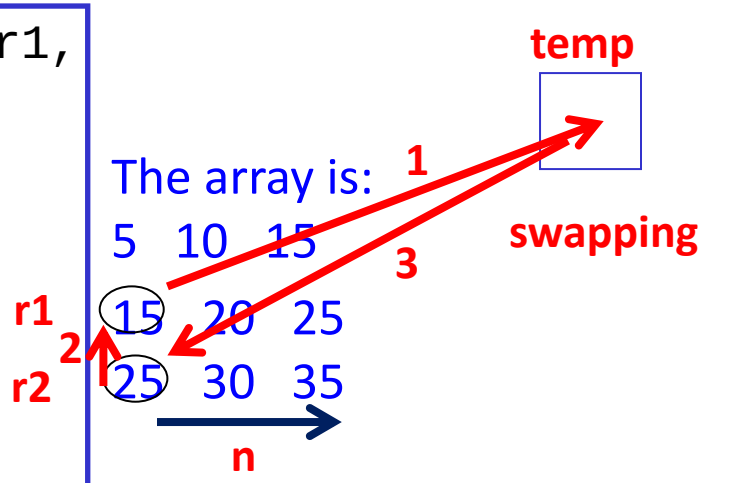
# Arrays – Q2

```
        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]);
      printf("\n");
   }
}
```

# Arrays – Q2

```
void swap2Rows(int M[SIZE][SIZE], int r1,
 int r2)
/* swaps row M[r1] with row M[r2] */
{
   int temp;
   int n;     // variable for column

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




The array is:

5 10 15

15 20 25

25 30 35

Enter two rows for swapping:

***1 2***

The new array is:

5 10 15

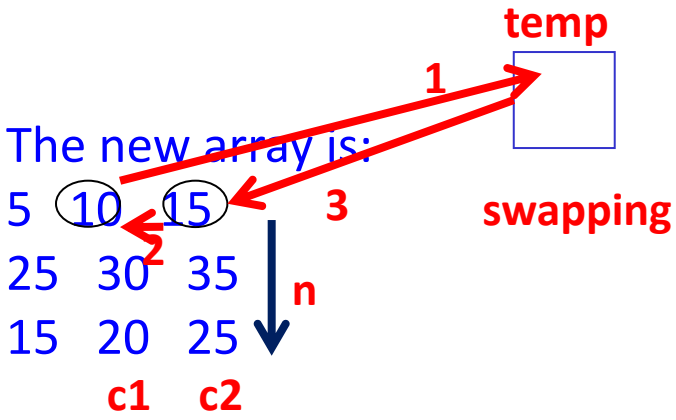
**25 30 35**

**15 20 25**

**Note: For the specified rows or columns, just perform a swapping operation for each element of the rows and columns**

# Arrays – Q2

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

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




Enter two columns for swapping:

***1 2***

The new array is:

5 **15** **10**

25 **35** **30**

15 **25** **20**

**Note: For the specified rows or columns, just perform a swapping operation for each element of the rows and columns**

# Arrays – Q3

(**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().

(1)One version printReverse1() uses index notation.

(2)The other version printReverse2() uses pointer notation for accessing the element of each index location.

In addition, write another C function **reverseAr1D()** that takes in an array of integers **ar** and a parameter **size** that 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);**

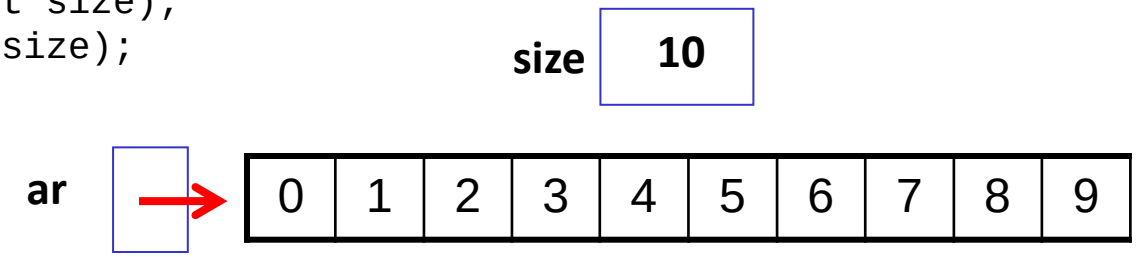
Write a C program to test the functions.

**Sample input and output session:**

```
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
```

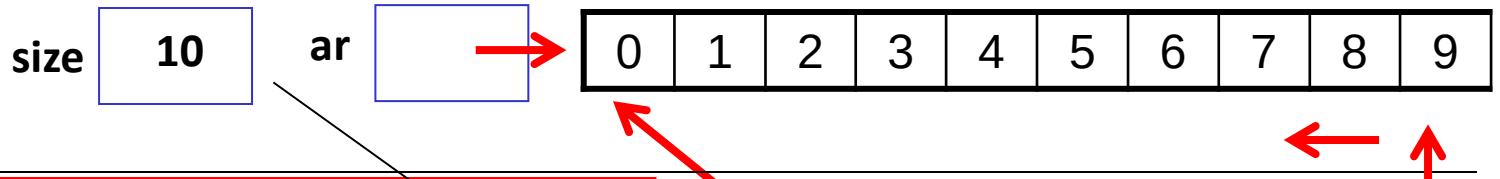
# Arrays – Q3

```
#include <stdio.h>
int readArray(int ar[ ]);
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;
}
```

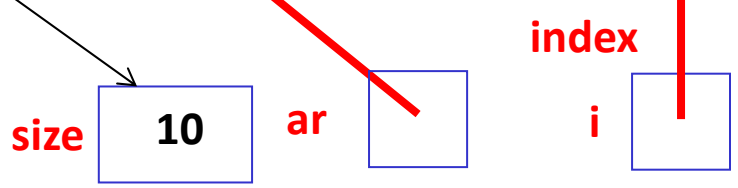


# Arrays – Q3

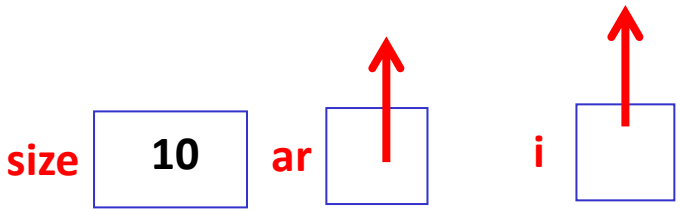
```
int main(){

}
```



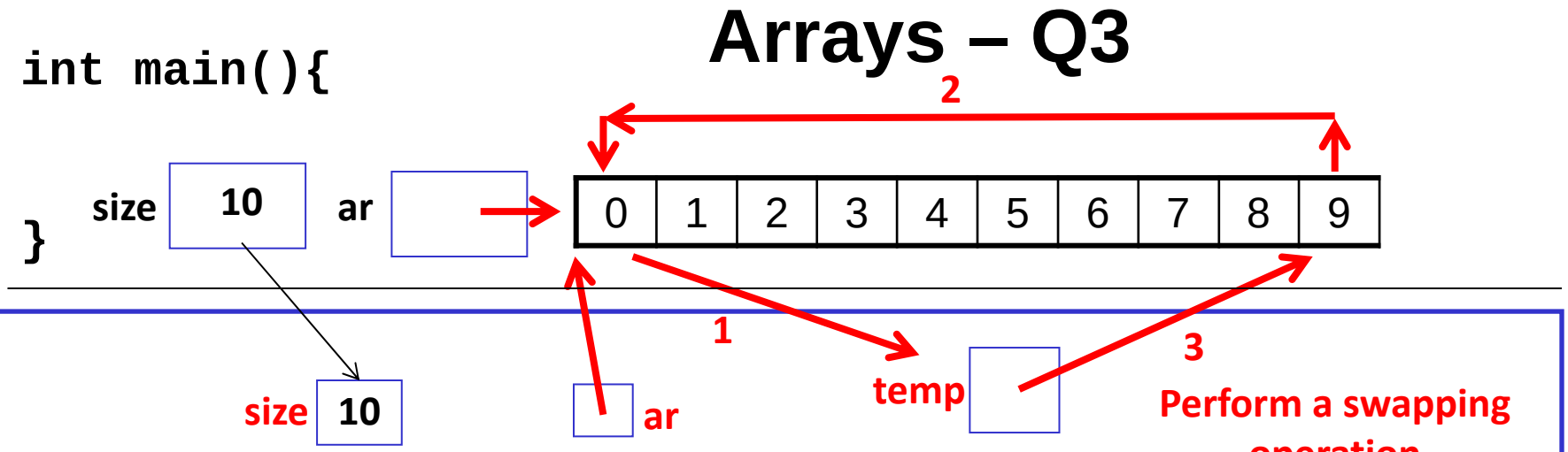

```
void printReverse1(int ar[ ], int
size){
   int i;
   printf("printReverse1(): ");
   if (size > 0) {
      for (i=size-1; i>=0; i--)
         printf("%d ", ar[i]);
   }
   printf("\n");
}
```




```
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");
}
```



# Arrays – Q3




```
/* reverseAr1D 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;
      }
   }
}
```

**Using Call by Reference**
**- No return statement**