WEEK 4 LAB - ARRAYS

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:

```
#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 program 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. (**findMinMax2D**) Write a C function that takes a 5x5 two-dimensional array of integers `ar` as a parameter. The function returns the minimum and maximum numbers of the array to the caller through the two pointer parameters `min` and `max` respectively. The function prototype is given as follows:

```c
void findMinMax2D(int ar[SIZE][SIZE], int *min, int *max);
```

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

```c
#include <stdio.h>
#define SIZE 5
void findMinMax2D(int ar[SIZE][SIZE], int *min, int *max);
int main()
{
   int A[5][5];
   int i,j,min,max;

   printf("Enter the matrix data (%dx%d): \n", SIZE, SIZE);
   for (i=0; i<5; i++)
      for (j=0; j<5; j++)
         scanf("%d", &A[i][j]);
   findMinMax2D(A, &min, &max);
   printf("min = %d\nmax = %d", min, max);
   return 0;
}
void findMinMax2D(int ar[SIZE][SIZE], int *min, int *max)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the matrix data (5x5):
1 2 3 4 5
2 3 4 5 6
4 5 6 7 8
5 4 23 1 2
1 2 3 4 5
min = 1
max = 23
```

(2) Test Case 2:

```
Enter the matrix data (5x5):
1 2 -3 4 5
2 3 4 5 6
4 5 6 7 8
```

```
5 4 23 1 27
1 2 3 4 5
min = -3
max = 27
```

3. (**diagonals2D**) Write a C function that accepts a two-dimensional array of integers ar, and the array sizes for the rows and columns as parameters, computes the sum of the elements of the two diagonals, and returns the sums to the calling function through the pointer parameters, `sum1` and `sum2`, using call by reference. For example, if the `rowSize` is 3, `colSize` is 3, and the array ar is {1,2,3, 1,1,1, 4,3,2}, then `sum1` is computed as 1+1+2=4, and `sum2` is 3+1+4=8. The function prototype is given as follows:

```
void diagonals2D(int ar[][SIZE], int rowSize, int colSize, int
*sum1, int *sum2);
```

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

```
#include <stdio.h>
#define SIZE 10
void diagonals2D(int ar[][SIZE], int rowSize, int colSize, int
*sum1, int *sum2);
int main()
{
   int ar[SIZE][SIZE], rowSize, colSize;
   int i, j, sum1=0, sum2=0;

   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]);
   diagonals2D(ar, rowSize, colSize, &sum1, &sum2);
   printf("sum1=%d; sum2=%d\n",sum1,sum2);
}
void diagonals2D(int ar[][SIZE], int rowSize, int colSize, int
*sum1, int *sum2)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter row size of the 2D array:
3
Enter column size of the 2D array:
3
Enter the matrix (3x3):
1 2 3
1 1 1
4 3 2
sum1=4; sum2=8
```

(2) Test Case 2:
```
Enter row size of the 2D array:
4
Enter column size of the 2D array:
4
Enter the matrix (4x4):
1 2 3 4
```

```
1 1 2 2
2 2 1 1
5 4 3 2
sum1=5; sum2=13
```

(3) Test Case 3:

```
Enter row size of the 2D array:
5
Enter column size of the 2D array:
5
Enter the matrix (4x4):
1 2 3 4 1
1 1 2 2 1
2 2 1 1 1
5 4 3 2 1
5 4 3 2 1
sum1=6; sum2=13
```