Data Structures and Algorithms

Pierre Collet/ Manal ElZant-Karray

Practical work n°2 : Arrays/Sort

Exercise 1: There are two sorted arrays: the first one is of size (m+n) containing only m elements and the second one is of size n and contains n elements.

Merge these two arrays into the first array of size (m+n) such that the output is sorted.

Use this sample:

1	EMP	5	EMP	EMP	8	EMP

Input: the array **mPlusN[]** with m+n elements.

EMP = Value is not filled/available in the array mPlusN[] (There should be n such array blocks).

Input: the array **N[]** with n elements.

0	4	7	10

Output: **N[]** merged into **mPlusN[]** (Modified mPlusN[])

0	1	4	5	7	8	10

Exercise 2: Write a C program to left rotate an array by **n positions**.

Use this sample:

Read the array (arr[]):

1	2	3	4	5	6	7	8	9	10

Read number of times to rotate N: 3

Left rotation is shifting of array elements to one position left and copying first element to last.

Output:

4	5	6	7	8	9	10	1	2	3

Exercise 3: Write the **QuickSort** function in the following C program to sort in a ***descending order*** the array a[].

```
#include <stdio.h>
#include <stdlib.h>
void QuickSort(……)
    {
…….
    }

int main() {
  int i;
  int a[8] = {20, 3, 45, 500, 22, 13, 120, 95};
  QuickSort(…);
  for (i = 0; i < 8; i++) printf("%d ,",a[i]);
  printf("\n");
   return 0;
}
```

Exercise 4: Predict the output of the following programs:

```
#include <stdio.h>
void nicerec(int n) {
  if(n > 0) {
    nicerec(n-1);
    printf("%d ", n);
    nicerec(n-1);
  }
}
int main() {
  nicerec(4);
  return 0;
}
```

//----------

```
#include <stdio.h>
int f(int n)
{
    if(n <= 1)
        return 1;
    if(n%2 == 0)
        return f(n/2);
    return f(n/2) + f(n/2+1);
}
int main()
{
    printf("%d", f(11));
    return 0;
}
```