MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG140 C Programming – Worksheet 14

Purpose:
In this work sheet, you will use arrays to implement one basic sorting and one searching algorithm.

Tasks:

a. There are different **sorting** algorithms. In this first task, write a C program that implements the following algorithm known as *bubble sort*.

"Bubble sort is a simple sorting algorithm. It works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list".

Consider following pseudo code while writing your program:

```
algorithm bubbleSort
    Initialize A[n]
    do
        swapped = false
        for i = 1 to n - 1 inclusive do
            if A[i - 1] > A[i] then
                /* swap them and remember something changed */
                swap(A[i - 1], A[i])
                swapped = true
            end if
        end for
    while(swapped)
end algorithm
```

A sample run would be as follows for an array [5 1 4 2 8]:

```
After Pass 1: [1 4 2 5 8]
After Pass 2: [1 2 4 5 8]
After Pass 3: [1 2 4 5 8]
```

b. A common requirement for most programs is to **search** a list for a given element. The two most common methods of performing such searches are linear and binary search algorithms. In this second task, write a C program that finds an element in a sorted array using binary search.

"Binary search inspects the middle element of the sorted list: if equal to the sought value, then the position has been found; otherwise, the upper half or lower half is chosen for further searching based on whether the sought value is greater than or less than the middle element."

A sample run would be as follows for an array [23 78 45 8 32 56]:

```
Enter a value to search: 45
This value is in the array!
```

A sample run would be as follows for an array [23 78 45 8 32 56]:

```
Enter a value to search: 100
This value is NOT in the array!
```

c. Write a complete C program for the problem definition below. Make sure that you make use of appropriate types of arrays, label the output clearly, and make sure that you make use of efficient control structures. Suppose you are given a table of integers, A, having m rows and n columns, and a list of integers, X, having n elements. We wish to generate a new list of integers, Y, that is formed by carrying out the following operations:

Y[1] = A[1][1]*X[1] + A[1][2]*X[2]+……+A[1][N]*X[N]
Y[2] = A[2][1]*X[1] + A[2][2]*X[2]+……+A[2][N]*X[N]
…..
Y[M] = A[M][1]*X[1] + A[M][2]*X[2]+……+A[M][N]*X[N]

Write a program that initialises A and X as below and generates Y output.

$$A=\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \end{pmatrix}$$

$$X=\begin{pmatrix} 1 \\ -8 \\ 3 \\ -6 \\ 5 \\ -4 \\ 7 \\ -2 \end{pmatrix}$$

$$Y=\begin{pmatrix} 4 \\ 0 \\ -4 \\ -8 \\ -12 \\ -16 \end{pmatrix}$$

d. Digital images represented with pixels (numbers) of 2D arrays. Write a program to create a digital image (2D array) with size 5x5 and randomly assigned pixels (numbers) between 0 and 255. And also create a mask (which is a 2D array) with size 5x5, pixels at (1,1),(1,2), (1,3), (2,1),(2,2), (2,3), (3,1),(3,2), (3,3) will be set to 1 and remaining will be set to 0. Then, write a function "masking" which will simply multiply the image with the mask.

A sample run would be as follows:

```
Created image is as follows:
120   100    50   200   220
120   100    55   200   220
 50    80    48    25   245
250    20   252    45   120
198    10    10    30    90

Mask is as follows:
0     0     0     0     0
0     1     1     1     0
0     1     1     1     0
0     1     1     1     0
0     0     0     0     0

Masked image is as follows:
0     0     0     0     0
0   100    55   200     0
0    80    48    25     0
0    20   252    45     0
0     0     0     0     0
```

e. Recently, the hospital Manager has been receiving patient feedback saying that the wait time for a patient to be served by a doctor are too long. Hence, Manager would like to see and analyse bins of patients waiting time. Write a program to create an array and populate it with random waiting time between 0 - 60 minutes for 20 patients. Then, write a function "bins" which will take this array as an input, evaluates and creates an array of bins with 5 minutes intervals of waiting time (12 intervals), and display the array content. In your program define the sizes of arrays as a symbolic constant called N.

Function prototype: void bins (int [N]);

A sample run would be as follows:

```
Patient's Waiting times:
5 36 8 5 4 60 7 15 25 36 38 57 56 2 0 3 6 3 10 51

Histogram of patient's waiting times:
  0-5 minutes: 7
 6-10 minutes: 4
11-15 minutes: 1
16-20 minutes: 0
21-25 minutes: 1
26-30 minutes: 0
31-35 minutes: 0
36-40 minutes: 3
41-45 minutes: 0
46-50 minutes: 0
51-55 minutes: 1
56-60 minutes: 3
```

Recommended Reading: Chapter 8 (p. 375-440)
Recommended Exercises: All programming exercises in Chapter 8.