COMP 1602 – Computer Programming II

2023/2024 Semester 2

Lab #11

Linear Search and Binary Search
2-D Arrays (Review)

1. An integer array, *num*, is sorted in ascending order. It contains the following values:

0	1	2	3	4	5	6	7	8	9	10	11
23	36	45	55	68	79	81	90	102	116	124	135

a) How many searches are required to find 90 using:
   i. binary search?
   ii. sequential search?

b) How many searches are required to find 60 using:
   i. binary search?
   ii. sequential search?

c) Suppose the array was not sorted. How would this affect the number of searches that are required for a sequential search?

d) What statement can you make about the relative efficiency of the binary search algorithm compared to the sequential search algorithm on a sorted array?

e) Identify the change/s that must be made to the binary search algorithm if the array is sorted in *descending* order. Write the code for the *binarySearch* function.

2. An integer array, *arr*, contains the following data:

0	1	2	3	4	5	6	7	8	9	10	11
23	6	25	15	8	79	81	90	1	56	32	101

a) Using binary search with the assumption that the array is sorted in ascending order, how many searches are required to find 23 and 32?

b) Sort the array in ascending order (manually).

c) Using binary search on the sorted array, how many searches are required to find 23 and 32?

d) Are the results from (c) the same as what you obtained in (a)?

3. A string array *A* contains the following values:

A	0	1	2	3	4	5	6	7	8	9	10
	Danny	Brandon	Barry	Yara	Mary	Shaun	Amelia	Carrie	Xia	Carson	Zack

a) The *binary search* algorithm is used to search the array *A* for the value “Mary”. Draw a trace table which shows the values of *low*, *high* and *mid* in searching for “Mary”. Will the *binary* search find Mary?

b) If *all* the criteria for the binary search were met by the array *A*, how many comparisons will be made before “Mary” was found?

4. a) Figure 1 below shows a two-dimensional array of integers, ***A***:

0	1	2	3	4	5
1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10

Figure 1

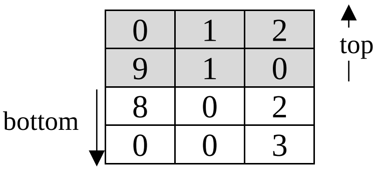
i) Write code to declare the array ***A***.

ii) The array shown in Figure 1 has already been loaded with values. Write a segment of code that can be used to assign the values shown in Figure 1 to the array. You **must** use nested '*for*' loops in your answer.

iii) Write a segment of code to interchange the values in column 0 with the values in column 5.

iv) Write a segment of code to *randomly* generate a location in the array and assign the value of -1 to that location in the array.

b) A 2D array is called *top heavy* if the top half of the array contains *more* **non-zero** elements than the bottom half of the array. The 2D array must have an even number of rows. For example, the following 2D array is top heavy:



0	1	2
9	1	0
8	0	2
0	0	3

Write a function, *topHeavy*, which accepts a 2D array ***m*** with *numRows* rows and *numCols* columns as parameters (where the maximum number of rows and columns is 100), and returns *true* if ***m*** is top heavy. The function must return *false* if ***m*** is not top heavy or *numRows* is odd.

End of Lab #11