COMP 5481

LAB 2 – Debug Search

Problem Description:

The concept of a binary search is fairly straight forward. Given a collection, firstly you must sort it then start in the middle of the collection, dividing it in two. Once you have divided it, you look to see if the number you're looking for is the middle number. If it is then you have found the number; if not you check whether the number is smaller or greater than the middle number. If it's less than the middle number, you search only in the first half. If it's greater, you search only in the second half. Repeat the same routine until you either find the number or there is only one number left in the collection which is not the one you are looking for.

For example if the collection is {2, 4, 5, 3, 1, 7, 6, 9, 0} and the number to search for is 3. First we will sort the collection C and we get C = {0, 1, 2, 3, 4, 5, 6, 7, 9}. LI ➔ LowerIndex, UI ➔ UpperIndex, Num ➔ Number to Search

Iteration	LI, UI	Mid, Val	Condition	Output
1	0, 8	(0+8)/2=4, 4	3 < 4	-1
2	0, 3	(0+3)/2=1, 1	3 > 1	1
3	2, 3	(2+3)/2=2, 2	3 > 2	1
4	3, 3	(3+3)/2=3, 3	3 == 3	found

At the end of iteration 4, we find the number 3 and we print "found". If we can't find the number at the end, we print "notfound".

Input:

Input to the program is given in two lines with line one containing the collection and line two containing the number to search for. The numbers on the first line are all integers between 0 and 500 both inclusive separated by a single space. You can assume that this collection will contain unique values and its length *len* will be 1<*len*<50.

Output:

The output should be one line where you print sequence of -1 and 1 depending on the iterations followed by "found" or "notfound".

SR. No	Input	Output
1	2 4 5 3 1 7 6 9 0 3	-1 1 1 found
2	12 54 25 13 31 3 16 69 100 29 36 13	-1 found
3	2 5 25 3 1 6 9 7 18	1 1 1 -1 notfound