CST 370 Spring 2017 : Final Exam (Take-home projects)

Name: Mariya Eggensperger
Email: meggensperger@csumb.edu

Important Note:

- There are two projects, each worth 50 points, total 100 points
- This is take-home exam, open book, open notes, and open computer, but you MUST work independently

Project #1 (50 points). Consider that a sorted list of integers (e.g., 1, 1, 2, 2, 5, 5, 5, 5, 5, 5, 5, 8, 8, 9, 10, 11, 11) provided to you. The goal is to find the occurrences of an integer in the array using binary search. For the above example, the occurrence of integer "5" should be 7.

a) (15 points) Design the algorithm and describe it in Pseudo code, including necessary explanations so that the logic of your algorithm can be understood.

<table>
<tr><th>PSEUDOCODE</th><th>LIVE CODE</th></tr>
<tr><td><pre>
Procedure: Binary Search
A ← sorted array
size_arra_A ← size of array
k ← value to be searched
Is_found ← boolean for if the value to be
searched is found

Set noElementFound = -1
Set lowerBound = 0
Set upperBound = size_arra_A-1;

while lowerBound<=upperBound
set midPoint = lowerBound + (upperBound -
lowerBound) / 2
 if A[midPoint] == search element
    Return = middle element
if (the search element is found)
      set upperBound = midPoint - 1
Else
      set lowerBound = midPoint + 1

Else if
    The search element is less than the middle
array element
        set upperBound = midPoint - 1
Else
    set lowerBound = midPoint + 1

Or no element found

      end while

end procedure
</pre></td><td><pre>
int binary_search(int Arr[],int size_arra_A,int k,bool
is_found) {

 int no_elem_found= -1;

 int low=0,high=size_arra_A-1;

 while(low<=high){

  int mid=(low+high)/2;

  if(Arr[mid]==k) {
      no_elem_found=mid;

  if(is_found)
         high=mid-1;

  else
         low=mid+1;
    }

 else if(k<Arr[mid]) high=mid-1;

  else low=mid+1;
 }

 return no_elem_found;
}
</pre></td></tr>
</table>

b) (20 points) Implement your algorithm in C++ and test your implementation with a tester/driver. The sorted list is stored in an array. The test array may be hard coded in the tester program, but the integer number to be searched for should be input from user (on console). Special cases should be properly handled.

From the images, it is evident that the code stores the sorted list in an array. The test array is hard coded into the tester program, but the integer number is searched by the user via console. Special cases such as out of bounds numbers and non integer elements such as symbols are rejected:

Figure 1: program asks user to input integer to be found (int k).

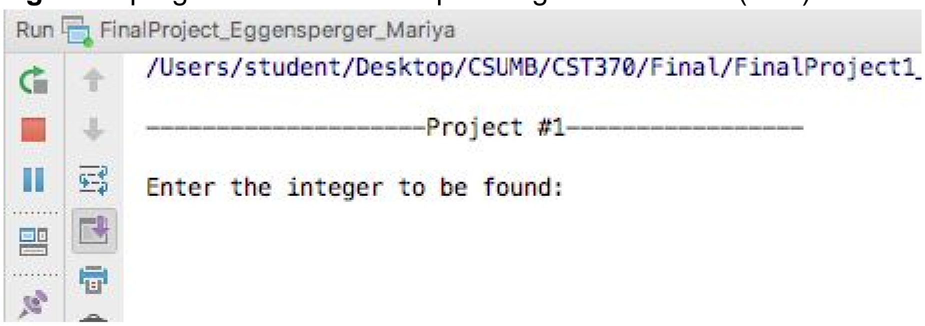


Figure 2: program provides user with first and last instance of the search value index occurrence. Then the program concludes the total occurrences of the number input.

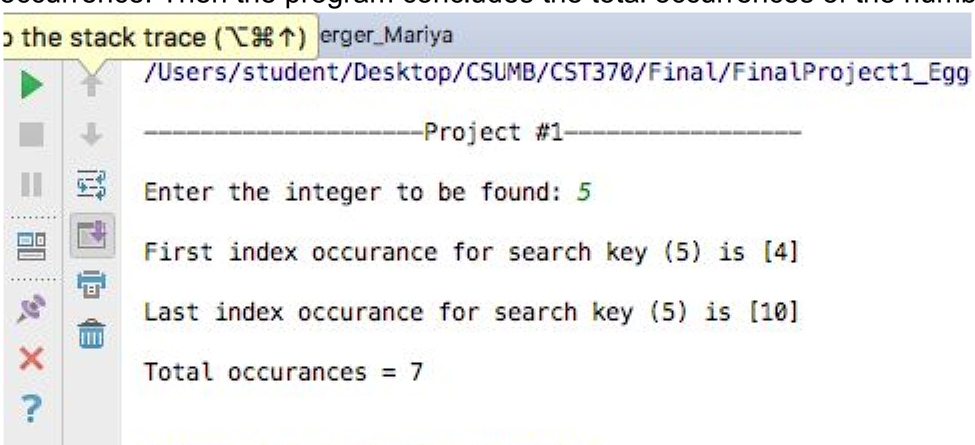
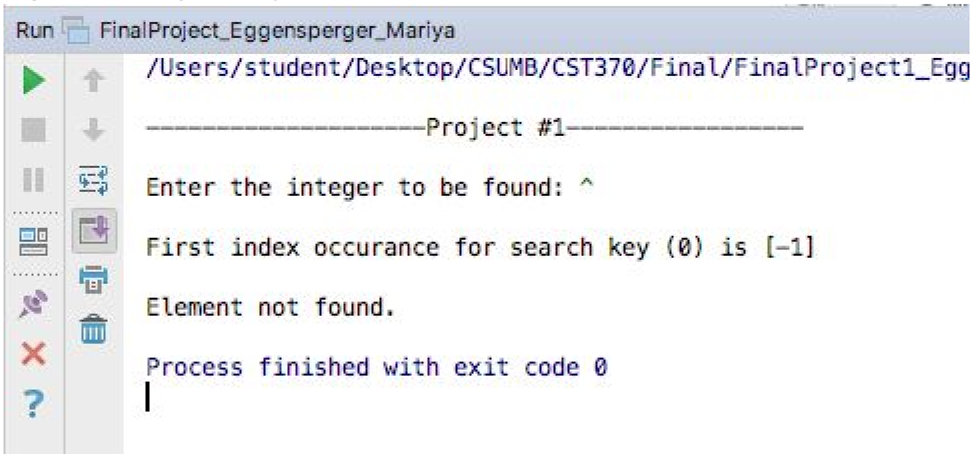


Figure 3: program rejects invalid tokens.



c) (10 points) Give the running time of your algorithm in big O notation (briefly explain your reasoning). If the running time of your algorithm is in O(log n) complexity, you will be given credit for this part (10 points).

The running time of my algorithm is in O(log n) complexity.

<table>
<tr>
<td>My program has a similar structure to this particular algorithmic proof. As is evidenced in this proof, Binary search is an efficient algorithm for finding an item from an ordered list of items. It works by repeatedly dividing in <u>half the portion</u> of the list that could contain the item, until the array keys are narrowed down to to just one target key.</td>
<td>With that said, binary search always works in O(log n) run time. Essentially, in this search algorithm, a problem of size n is divided into a subproblem of size n/2 until a conclusion is reached that the problem becomes of size 1. See similarities in algorithm structure.</td>
</tr>
<tr>
<td>
<pre>
BINSEARCH (A, x, a, b)
    if b = a then
        return false
    m ← (b−a)/2 + a
    if A[m] > x then
        return BINSEARCH (A, x, a, m)
    else if A[m] = x then
        return true
    else if A[m] < x then
        return BINSEARCH (A, x, m, b)
</pre>
</td>
<td>
<pre>
*/
int binary_search(int Arr[],int size_arra_A,int k,bool is_found){
   int no_elem_found=-1;
   int low=0,high=size_arra_A-1;
   while(low<=high){
      int mid=(low+high)/2;
      if(Arr[mid]==k) {
         no_elem_found=mid;
         if(is_found)
            high=mid-1;
         else
            low=mid+1;
      }
      else if(k<Arr[mid]) high=mid-1;
      else low=mid+1;
   }
   return no_elem_found;
}
</pre>
</td>
</tr>
</table>

d) (5 points) Make a video to give an overview of your algorithm and implementation. Include the link to the video in your written document.

Video Link: https://youtu.be/IOmYRzczruI

Submission instruction: Zip your source programs and written document in a single file named as ‘Final_Project1_yourlastname’. For the program, please include only the source files needed to compile and run successfully.