Instructor Name: Siddhartha Nath
Email: s.nath@northeastern.edu

Northeastern University

Program Structure and Algorithms (INFO 6205)
Quiz #3 – SAMPLE SOLUTIONS – 30 points

Student NAME:

Student ID:

Question 1 (30 points)**.** *You are given a* **sorted** *array of integers* $A[1:n]$ *that may or may not contain duplicate values. You are also given a* **target** *integer* k*. You want to find out if there strictly more than one occurrence of* k *in* $A[1:n]$ *and the count of the occurrence.*

For example, if $A = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]$ *and* $k = 5$*, then your algorithm should output* 3*. If* $k = 3$ *or* $k = 7$*, then your algorithm should output "False".*

(a) *(2 points) Please describe a linear search algorithm* **in English**.

Run a linear search on $A[1:n]$ *and count the number of occurrences of* k*. If the count is* > 1*, the output the count, else "False". We can use a dictionary / hash table for keep track of the counts per key, which are each unique values in* $A[1:n]$*.*

(b) *(2 points) What is the asymptotic running time of your algorithm in (a) in* $O(\cdot)$ *or* $\Theta(\cdot)$*?*

Worst-case running time is $\Theta(n)$*.*

(c) *(6 points) Please describe an efficient divide-and-conquer algorithm* **in English** *to find the first and last occurrences of* k*.*

We can modify the D/Q algorithm binary search. Since $A[1:n]$ *is sorted, all occurrences of* k *will be adjacent. We need to find the indices of the first (*i*) and last (*j*) occurrences of* k*. The count will be* $j - i + 1$*.*

To find the first index, i*, we recursively search for* k *in the array, creating only one subproblem for either* $k \leq A[mid]$ *or larger. That is, we can discard half of the input problem size.*

To find the last index, j*, we recursively search for* k *in the array, creating only one subproblem such that* $k > A[mid]$ *or* $\leq$*.*

(d) *(6 points) Please write the* **pseudocode** *of your divide-and-conquer algorithm in (c). You must use recursion to receive any credit.*

```
procedure find_first_index(A, start, end, k)
mid ← (start+end)//2
if end >= start:
  if (A[mid] == k) and (mid == 0 or k > A[mid − 1]):
    return mid
```

```
  else if k ≤ A[mid]:
    return find_first_index(A, start, mid-1, k)
  else:
     return find_first_index(A, mid+1, end, k)
return -1
```

```
procedure find_last_index(A, start, end, k)
mid ← (start+end)//2
if end >= start:
  if (A[mid] == k) and (mid == end or k < A[mid + 1]):
    return mid
  else if k < A[mid]:
    return find_last_index(A, start, mid-1, k)
  else:
     return find_last_index(A, mid+1, end, k)
return -1
```

```
procedure count_occurrence(A, k)
first ← find_first_index(A, 1, n, k)
if first != -1:
  last ← find_last_index(A, first, n, k)
  count ← last - first + 1
  print(count)
else:
  print(False)
```

(e) *(3 points) Please write the* **recurrence relation** *($T(n)$) of your pseudocode in (d). That is, $T(n) =$???.*

In each recursion, we create one subproblem that is half the size of the original problem. There is no overhead of splitting or merging the solutions. Therefore, $T(n) = T(n/2) + \Theta(1)$.

(f) *(3 points) Please solve your recurrence in (e) using the Master method. Please clearly write the asymptotic running time of your algorithm in $O(\cdot)$ or $\Theta(\cdot)$.*

$a = 1, b = 2, d = 0, k = 0, f(n) = \Theta(1) \Rightarrow d = 0 = log_b a \Rightarrow$ *Case 2, so* $T(n) = \Theta(\log n)$.

(g) *(8 points) For the example, $A = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]$ and $k = 5$. please fill in the tables below for the values of start, end, mid, returned indices.*

find_first_index() calls	*start*	*end*	*mid*	*Returned Index (ϕ if recursion continues)*
1	*1*	*10*	*5*	ϕ
2	*1*	*4*	*2*	*2*
3				

find_last_index() calls	*start*	*end*	*mid*	*Returned Index (ϕ if recursion continues)*
1	*2*	*10*	*6*	ϕ
2	*2*	*5*	*3*	ϕ
3	*4*	*5*	*4*	*4*