NOTE TO STUDENTS: This file contains sample solutions to the term test together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

Question 1. [12 MARKS]

Consider the following algorithm that finds the *last* occurrence of an element x in an array A.

```
FINDLAST($A, x$):
    $i \leftarrow A.length - 1$
    while $i \geqslant 0$ and $A[i] \neq x$:
        $i \leftarrow i - 1$
    return $i$
```

For this question, we measure complexity by counting only the number of array accesses (like in Question 1 of Assignment 1).

Part (a) [2 MARKS]

What is the worst-case complexity of FINDLAST? Give an **exact** expression and justify your answer.

SAMPLE SOLUTION:

Worst-case number of array accesses = n: no element is accessed more than once (so worst-case $\leqslant n$) and when x does not appear in A, every element is accessed at least once (so worst-case $\geqslant n$).

MARKING SCHEME:
- [1 mark] correct answer
- [1 mark] good justification

Part (b) [10 MARKS]

What is the average-case complexity of FINDLAST? Clearly define your sample space and probability distribution—assume that $\Pr[x \notin A] = 1/3$—explain what you are doing, show your work, and simplify your answer.

SAMPLE SOLUTION:

Sample space: $S_n = \{(A,x) : A = [1,\dots,n], x = 0,1,\dots,n\}$—one input for every possible position of x in a list of size n, plus one more for the case when $x \notin A$.
Probability distribution: $\Pr[x=0] = 1/3$ (as given in the question); $\Pr[x=i] = 2/3n$ for $i = 1,\dots,n$ (all other inputs equally likely).

Let X = total number of array accesses. Then, $X = n$ iff $x = 0$ or $x = 1$, and for $j = n-1,\dots,1$, $X = j$ iff $x = n+1-j$. Hence,

$$\begin{aligned}
E[X] &= \sum_{j=1}^{n} j \Pr[X=j] = \left(\sum_{j=1}^{n-1} j \Pr[x = n+1-j]\right) + (n \Pr[x = 1 \lor x = 0]) \\
&= \sum_{j=1}^{n-1} j \cdot \frac{2}{3n} + n\Pr[x=1] + n\Pr[x=0] = \frac{2}{3n}\sum_{j=1}^{n-1} j + n\frac{2}{3n} + n\frac{1}{3} \\
&= \frac{2}{3n}\frac{n(n-1)}{2} + \frac{n+2}{3} = \frac{2n+1}{3}.
\end{aligned}$$

MARKING SCHEME:
- [3 marks] sample space and probability distribution
- [3 marks] random variables and relationship to sample space
- [2 marks] clear attempt to compute the correct expected value
- [2 marks] calculation and algebra

ERROR CODES:
E1: missing sample space
E2: forgetting the case $x \notin A$
E3: missing or incorrect random variables
E4: algebraic error
E5: computing the wrong quantity
E6: missing "$\frac{2}{3n}$" in probability distribution
E7: did not define probability distribution

Question 2. [10 MARKS]

Recall that the *rank* of an element in a collection is its position in the sorted order of all the elements (where counting starts at 1). When duplicate values are allowed, the same value can have many different ranks. For example, in the collection $\{5.5, 1.3, 1.3, 8.9, 0.5, 3.4, 0.5\}$, the rank of 0.5 is both 1 and 2, while 3.4 has rank 5.

Consider the following "k-SET" ADT (for fixed $k \in \mathbb{Z}^+$).

Objects: A collection C of rational numbers (duplicate values are allowed).

Operations:

- INSERT(C,x): add element x to collection C.
- k-SELECT(C): return the element at rank k in C—or NIL if $|C| < k$.

Describe an implementation of the k-SET ADT, based on data structures covered in class. State the worst-case running time of each of your operations. *For full marks, your operations must be as efficient as possible—in particular, k-SELECT must run in worst-case time $\mathcal{O}(1)$.*

Sample Solution:

Store the k smallest elements in a *max*-heap H_1 and the remaining elements in a *min*-heap H_2—when $n < k$, $H_2.size = 0$ and $H_1.size = n$. We use indexing to access the root element, e.g., $H_1[1]$.

```
k-Select(C):
    if $H_1.size < k$:
        return nil
    else:
        return $H_1[1]$

Insert(C, x):
    if $H_1.size < k$ or $x < H_1[1]$:
        Heap-Insert($H_1, x$)
    else:
        Heap-Insert($H_2, x$)
    if $H_1.size > k$:
        Heap-Insert($H_2$, Extract-Max($H_1$))
```

Running time is $\Theta(1)$ for k-Select (no loop or recursive call) and $\Theta(\log n)$ for Insert (at most three heap operations performed).

Note that the min-heap H_2 is not strictly necessary for this problem; any simple container would work, e.g., an unsorted list.

Marking Scheme:

- [2 marks] description of data structure (how to store the elements)
- [3 marks] correctness and efficiency of algorithm for k-Select
- [1 mark] runtime analysis for k-Select
- [3 marks] correctness and efficiency of algorithm for Insert
- [1 mark] runtime analysis for Insert

Common Errors:

- Many students tried to solve this in complicated ways that did not achieve the required constant running time for k-Select: AVL trees, sorted lists, etc.

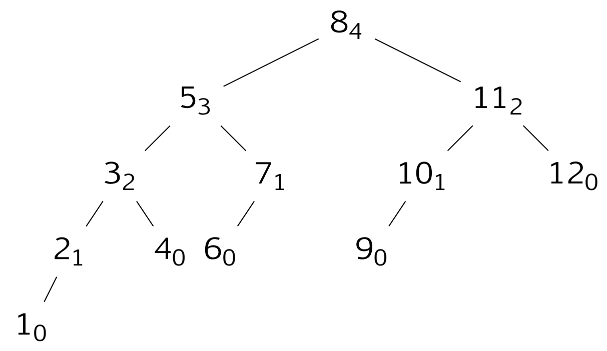
Question 3. [12 marks]

Part (a) [4 marks]

Draw an AVL tree containing items $\{1,2,3,4,5,6,7,8,9,10,11,12\}$, but make your tree *as unbalanced as possible*. Next to each node, indicate the height of the subtree rooted at that node.

Sample Solution:

Height is indicated as a subscript next to each item.

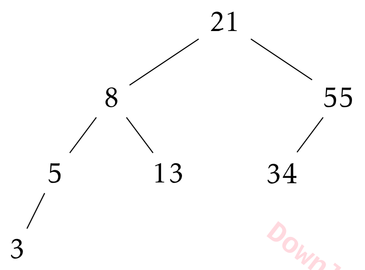


MARKING SCHEME:

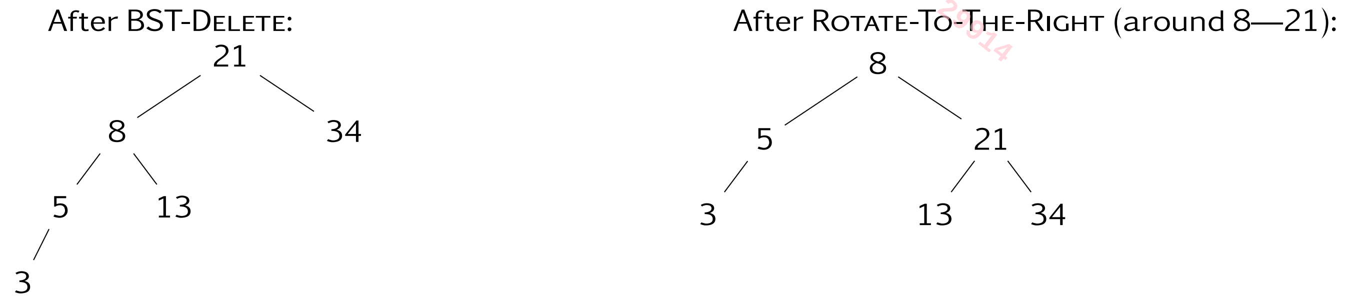
- [1 mark] tree is a BST
- [1 mark] correct heights are indicated
- [1 mark] tree is AVL-balanced
- [1 mark] tree is as unbalanced as possible for an AVL tree

Part (b) [4 MARKS]

Draw the result when 55 is deleted from the AVL tree below. Show your work—in particular, identify clearly any rotation performed.



SAMPLE SOLUTION:



MARKING SCHEME:

- [1 mark] result is a correct BST
- [2 marks] result is a correct AVL tree
- [1 mark] rotation done correctly

Part (c) [4 MARKS]

Let $m(h)$ be the *minimum* number of nodes in any AVL tree of height h. Give a recurrence relation for $m(h)$, including appropriate base cases. Explain your recurrence briefly.

SAMPLE SOLUTION:

- $m(0) = 1$: height 0 requires one node
- $m(1) = 2$: height 1 requires root with one child
- $m(h) = 1 + m(h-1) + m(h-2)$, for $h \geqslant 2$: height h requires root with one child of height $h-1$; by AVL balancing condition, other child has height at least $h-2$

MARKING SCHEME:

- [1 mark] appropriate base cases
- [2 marks] correct recursive case
- [1 mark] good explanation

Question 4. [6 MARKS]

We want to keep track of student records in CSC 263 H1 (with 135 students currently enrolled). The "key" for each student will be their student number.

Part (a) [3 MARKS]

Consider using a hash table of size 11 with chaining. Assuming we use a hash function that satisfies *simple uniform hashing*, how many student records will we encounter, on average, when searching for a specific student? State clearly the general expression for your answer, then give the value for this particular example (you can give your final answer as a fraction).

SAMPLE SOLUTION:

> Average number of elements examined during a successful search = $\frac{\alpha+1}{2} = \frac{n/m+1}{2} = \frac{n+m}{2m}$.
>
> In this case, $n = 135$ and $m = 11$ so $\frac{n+m}{2m} = \frac{146}{22} = \frac{73}{11}$.

MARKING SCHEME:

- [2 marks] correct general expression (1/2 for using "$\alpha = n/m$")
- [1 mark] correct value for example

Part (b) [3 MARKS]

Consider using a different hash table with size 135 and the hash function $h(N)$ = sum of the individual digits of N. (For example, $h(1234567890) = 1+2+3+4+5+6+7+8+9+0 = 45$.) Is this a good hash function to use? Give at least two good reasons to support your answer.

SAMPLE SOLUTION:

> h is **not** reasonable:
>
> - for all 10-digit student numbers N, $0 \leqslant h(N) \leqslant 90$—so not all table locatiosn will be used;
> - $h(N_1) = h(N_2)$ for any two numbers N_1, N_2 that are permutations of each other—so we expect many collisions.

MARKING SCHEME:

- [2 marks] for each reasonable reason

Bonus. [4 MARKS]

Solve your recurrence from Question 3(c).

SAMPLE SOLUTION: *(No solution for bonus questions. . .)*

MARKING SCHEME:

- [1 mark] correct answer
- [3 marks] good derivation