# University of Waterloo
# CS240 Winter 2025
# Assignment 3

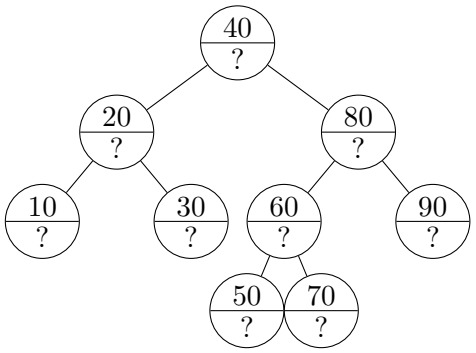
**Due Date: Tuesday, March 4 at 5:00pm**

Please read `https://student.cs.uwaterloo.ca/~cs240/w25/assignments.phtml#guidelines` for guidelines on submission. **Each question must be submitted individually to Crowdmark.** Submit early and often.

**Grace period:** submissions made before 11:59PM on March 4 will be accepted without penalty. Your last submission will be graded. Please note that submissions made after 11:59PM **will not be graded** and may only be reviewed for feedback.

## Question 1 [1+3+4=8 marks]

Consider the binary search tree $T$ on the right:




(a) Show that $T$ is an AVL-tree by writing the *balance* in the lower half of each node (Each answer must be in $\{-1, 0, 1\}$). No explanation needed.

(b) Show the result of performing insert(65) in $T$. You can just show the final tree, but there will be no credit if it is incorrect. If you want partial credit (in case the final tree is incorrect), then show the intermediate trees. You do not need to show the balance factors.

(c) List the largest key $k \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$ such that performing delete($k$) on the given tree $T$ would result in a rotation. Justify your answer by showing the tree just before the rotation is done, and after the rotation. Note that the given tree $T$ is the same tree used for part a and does not include the insert operation from part b.

## Question 2 [5 marks]

Let $T$ be an AVL tree. Let $L$ be a list containing the keys of $T$ in non-decreasing level order. That is, the root is the first element of $L$, the children of the root are the next two elements (note that the left child can be stored before the right child or the other way around), etc. Now suppose we build a new tree $T'$ by inserting elements of $L$ (in the order they appear in $L$) into an AVL tree. Is $T'$ an identical copy of $T$? Explain.

## Question 3 [4 marks]

We define a variant of AVL trees called *d-AVL trees* as follows. Let $d$ be a fixed constant. If the height of the node is greater than $d$, then the node must satisfy height-balance property. If the height of a node is less than or equal to $d$, then the node does not have to satisfy the height-balance property. Show that a d-AVL tree with $n$ nodes has height $O(\log n)$.

## Question 4 [4+2+2=8 marks]

**a)** Draw a diagram of a skip list starting with an empty skip list and inserting the seven keys $67, 28, 64, 66, 60, 81, 49$. Use the following coin tosses to determine the heights of towers (note, there may be tosses that are not used):

$$T, T, H, H, T, H, T, H, H, T, H, H, T, T, H, T, H, H, T, T, H, H, H, T, \ldots$$

It is sufficient to show the final skip list.

**b)** Suppose we use a biased coin to build a skip list, and the probability of $H$ (heads) is $1/3$. What is the probability that every tower in a skip list storing $n$ entries will have height strictly larger than 4?

**c)** Given items in a sorted array $A$ of size 13, we wish to store them in a skip list of a fixed height 2: $S_0$ stores all elements, $S_2$ stores only the sentinels. We can store any number of items we wish in $S_1$. Which items should we store in $S_1$ so that the worst case number of scan forwards is as small as possible? You can refer to the items using their indexes in array $A$. Only search operations are performed, i.e. no elements are inserted or deleted from the skip list. Also state the worst case number of scan forwards.

## Question 5 [2+2+(3+3)=10 marks]

**a)** Consider the list of keys:

[1 2 3 4 5 6 7 8 9 10]

and assume we perform the following searches:

10, 7, 2, 2, 4$^\star$, 1, 2, 1, 2, 1$^\star$, 1, 7, 1, 9$^\star$

Using the move-to-front heuristic, give the list ordering after each of the starred ($\star$) searches are performed.

**b)** Repeat part (a), using the transpose heuristic instead of the move-to-front heuristic.

**c)** You are given a dictionary $D$ implemented using the Transpose heuristic, with $n$ distinct keys, stored as an unordered linked list $L$. Suppose you want to perform a sequence of $\sqrt{n}$ searches on keys present in $D$ where the number of different keys searched for is $\sqrt{n}$. What is the best-case and worst-case runtime to perform all $\sqrt{n}$ searches? Use $\Theta$ notation. Explain your answer.

## Question 6 [2+4=6 marks]

**a)** Illustrate interpolation search on array $A = [0, 10, 100, 1000, 10000]$ and search key 100. To illustrate the search, it is enough to show the sequence of values computed for $m$.

**b)** Let us call an array $A$ to be *positive quadratic* from indices $l$ to $r$, if there are numbers $a > 0$ and $b$, s.t. $A[i] = ai^2 + b$ for all indices $i$, such that $l \leq i \leq r$. Redefine the formula for $m$ in the interpolation search so that if the input array $A$ is positive quadratic between indices $l$ and $r$, and if the search key is $k = A[t]$, with $l \leq t \leq r$, then the new formula for $m$ immediately outputs $t$. Note that your formula for $m$ must lead to a valid search for an arbitrary sorted array $A$, i.e. if the key is present in $A$ from indices $l$ to $r$, it will be found even with the new formula. You cannot change anything else for interpolation search, only the formula for $m$.