

Problem Set #1: Basic Data Structures

Handed out on Feb. 12, due on Feb. 26 at the beginning of class. Remember: write your own answers and use English or pseudocode when algorithms are requested. Late homeworks will not be accepted (turn in whatever you have).

Problem 0. (Lewis & Denenburg) A *checkerboard* is a 2-dimensional array in which only elements (i, j) for which $i + j$ is even are ever used. Indices run from 0 to $n - 1$ in both dimensions. Explain how to store a checkerboard in contiguous memory in a space-efficient way.

Problem 1. Suppose you are given a (strange) computer that can only perform the following instructions (in addition to **if** and **while**):

$S := \text{create_stack}$ makes a new stack S , and

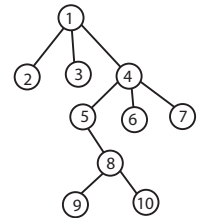
$i := S.\text{pop}$ removes the top item from stack S and places it in variable i , and

$S.\text{push}(i)$ makes item i the top item in stack S .

Solve the following problems:

- Show how you can use these operations to implement a queue (operations **makequeue**, **enqueue**, **dequeue**) — a picture might help to explain your answer.
- What's the worst case running time of your **dequeue** implementation?
- Over a series of n enqueues followed by n dequeues, how many **pop** operations does your implementation perform?

Problem 2. A *level order* traversal of a binary tree visits each node in increasing order of depth and from left to right within a level. The numbers in the figure at right give a level-order traversal. Give a (short) algorithm for performing a level-order traversal.



Problem 3. Describe how to reconstruct a binary tree if you are given *both* its preorder and inorder traversals. Is it possible given only a preorder traversal?

Problem 4. Suppose T_1 and T_2 are two *ordered, binary* trees. Let r_1 and r_2 be the roots of tree T_1 and T_2 , respectively, and denote the left and right children of a node u by $\text{LEFT}(u)$ and $\text{RIGHT}(u)$. Two such trees are *similar* if they have the same shape — in other words, if their natural drawings look the same. For example, (a) and (b) at right **are not** similar. Write recursive function to test whether two such trees T_1 and T_2 have the same shape.

