By signing my BannerID below, I affirm and certify that (1) I received no information from students who took the following exam early, and (2) I will give no information to students who are taking the exam at a later date.

**Banner ID (written legibly):**

Please do NOT put your name or login on this exam. Only Banner ID

This is the first exam for CS16. It starts at 7 PM and ends at 12 midnight. This exam is meant to take about 2-3 hours, but you may stay for up to 5 hours. We strongly recommend leaving by 11 PM; the chances are good that anything you add after that time will reduce rather than increase your grade.

- You may not use any resources except your mind and your writing implement (unless you have pre-discussed accommodations): no books, phones, computers, etc.

- You may ask the course staff for clarifications of problems.

Please do the following:

1. Read through the entire exam carefully, and make sure you understand what each question is asking. Please note the general guidelines on the next page.

2. You have plenty of time, so please write your answers clearly and neatly on the exam paper. For all problems, but especially for pseudocode, you may want to write a first and maybe even a second draft on a piece of scrap paper, and then write a clean copy on the exam.

3. We will only grade the work in this exam booklet or on scratch paper stapled to the exam.

4. Check your work carefully.

**Good luck!**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Possible Points</th>
<th>Score</th>
<th>TA Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
General assumptions and guidelines:

1. For pseudocode, **you may assume that your inputs are valid** (e.g. if we say that your function takes in an array of increasing numbers, you do not need to check that the array is not null or that the items are numbers or that they are increasing).

2. For pseudocode, you are free to write helper functions as needed. However, do not alter the input parameters that are stated for the functions given in the problem.

3. Unless specifically told otherwise, you may assume that you have access to general-purpose data structures and their methods that we have talked about in class. For example, if you would like to use a priority queue with $O(\log n)$ insert and removal of minimum element, you needn’t reimplement heap. You may simply create an instance of the data structure.

4. We will be grading based on correctness, clarity/succinctness, and efficiency. You could turn in perfectly-functional pseudocode that does not receive full credit because it is unreasonably cryptic or not the most efficient way to solve the problem.
The real test will consist of problems similar to some of the problems listed below and draw from the following topics. A really good way to study is to do all the problems below independently and then compare your solutions with a friend. If you actually solve the problems with friends, be sure you understand the solutions, and how to get to the solutions yourself. Memorizing the solutions to these problems will be of almost no value on the actual test. Make sure none of the following topics are foreign to you.

**Topics to Study**

Anything covered in lecture, projects, or homeworks before starting the graphs unit is fair game! Here is a list of topics you should be familiar with!

- Dynamic Programming
- Analysis and Big-O, Big-Omega, Big-Theta
- Amortized Analysis, Expected Analysis
- Expanding Data Structures (Stacks, Queues)
- Hashing, Sets, and Dictionaries
- Arrays, Binary Search
- Trees and Traversals
- Binary Search Trees
- Priority Queues, Heaps
- Sorting, Master Theorem
- Selection

**Topics You Are Not Responsible For**

- Hashing proof
- Proving expected runtimes (you should know *what* the expected runtimes of various algorithms/data structures are, but you will not need to prove that something has a certain expected runtime)

- Graphs
- DAGs and Topsort
- Decision Trees
1 Tree Traversal

A binary tree of height \( h \) can have at most \( 2^{h+1} - 1 \) nodes, as we showed in class. We define the fullness of a tree as the number of nodes in the tree divided by the maximum possible number of nodes for a tree of that height. Thus the tree below has a fullness of \( 4/7 \) (the height is 2, and there are 4 nodes, so the fullness is \( 4/(2^3 - 1) = 4/7 \)).

Design an efficient linear-time algorithm to compute, for any binary tree, the minimum fullness of any subtree of the tree (including the tree itself). In the example, there are four subtrees: the left leaf (fullness = 1), the right leaf (fullness = 1), the left subtree of the root (fullness 2/3) and the root (fullness = 4/7). The algorithm will return 4/7 because 4/7 < 2/3.
2 Pseudocode

Suppose you have a stack and you want to know whether it’s “almost empty”. Write pseudocode for a method that returns “true” if the stack is either empty or has just one item in it. The Stack class you’re enhancing does not keep track of its current size, \( n \), but has an \( isEmpty() \) function. Your method should be \( O(1) \), i.e., constant time.
3 Master Theorem

What is the big-O running time of a recursive algorithm that splits a problem of size $n$ into 3 subproblems each of size $2n/3$, recursively solves the three subproblems, and then combines the solutions in time $\Theta(n^2)$?

Recall the Master theorem: Let $T(n)$ be a monotonically increasing function that satisfies $T(n) = aT(n^{\frac{2}{3}}) + \theta(n^d)$. Then $T(n) =$

- $\Theta(n^d)$ if $a < b^d$
- $\Theta(n^d \log(n))$ if $a = b^d$
- $\Theta(n^{\log_a b})$ if $a > b^d$

Note: This will be provided on the midterm!
4 Sorting

You are given two arrays of integers, A and B, both sorted in increasing order. A has enough empty space at the end to include all the elements of B within it. Write pseudocode for a method to merge B into A such that all the elements in the merged array A are sorted in increasing order as well.

Example: A = [1,3,5,...] and B = [2,4,6] then your method should return the array [1,2,3,4,5,6,7]. Try hand simulating with your own examples.
5 Dynamic Programming

You’re given a rope of length n and list of prices of rope of length i where 1 <= i <= n. Find the optimal way to cut the rope into smaller ropes in order to maximize profit.

Example:
You’re given the array lengths[] = [1,2,3,4,5], prices[] = [1,5,8,9,10], and a rope length of 4. Here the maximized profit would be to cut the rope into two pieces of length 2 each for a profit of 10.

Write pseudocode for a way to solve this problem dynamically. A dynamic solution will break the large problem into smaller subproblems and then use the solutions to build up to solving the whole problem. Note: Refer to HW3 for more practice with dynamic programming