This is a set of practice questions for the final for CS16. The actual exam will consist of problems that are quite similar to those you have encountered on homeworks, the midterm, and on this practice sheet. Since you will not hand in solutions to these practice problems, you are welcome to work together on them, although we advise that you go ahead and write up solutions as a way of making certain that you really understand the answers found during discussion. Keep in mind that you are responsible for understanding the concepts behind any of the mentioned algorithms.
Note : This practice exam is longer than the actual exam would be so you should not worry if it takes you longer than three hours to attempt.

Login: ____________________

Problem 1
For each of the following, give a worst-case runtime, and a brief explanation (a sentence or two). In cases where the algorithm isn't specified, use the best one you can think of.

1. Removing the minimum element from a heap that contains n items
2. Constructing a minimum spanning tree for a connected graph with n nodes and n^2 edges using Kruskal's algorithm
3. Determining whether an item occurs in an unsorted singly-linked list
4. Sorting n integers, all between 0 and 9, where n is much greater than 100.
5. Finding a spanning tree (*not* necessarily a minimum-weight spanning tree) for a connected graph of n nodes and $n \log n$ edges.
6. Doing a pre-order traversal for a binary tree with n nodes.

Problem 2

In a binary search tree, T, in-order traversal will visit the nodes in increasing-key order, thus providing a linear-time mechanism for extracting all keys in order.

Suppose that you have some *range* of keys, say $k1 \leq k \leq k2$, and you'd like to extract all keys in T that lie within this range, and you'd like to get the results, as before, in increasing-key order. (Note that neither $k1$ nor $k2$ needs to actually be a key in the BST, and that the returned list of keys might well be empty!) Once again, inorder traversal with a test for in-range-ness will solve the problem in worst case $O(n)$ time, where n is the number of keys in the tree. You are to describe how to solve the problem in time $O(h + s)$, where h is the height of the tree T, and s is the number of keys returned.

- Give a brief description of your approach to this problem.
- Write pseudocode for your algorithm.
- Explain briefly why your algorithm is $O(h + s)$.

Problem 3

By now you should know how to construct a minimum spanning tree from a graph. However, sometimes we don't want a minimum spanning tree – perhaps we want to force our spanning tree to contain specific edges, regardless of whether or not those edges are optimal.

Suppose you are given an undirected graph $G = (V, E)$ with edge weights w_e, and an acyclic set of edges F, where F is a subset of E. (You may assume the edges are decorated such that "inF(e)" will return either true if e is one of the edges in F, or false otherwise.) Develop an algorithm that runs in $O(|E|\log|E|)$ and returns the lightest spanning tree T such that the edges of F are all contained in T.

- Give a brief description of your approach to this problem.
- Write pseudocode for your algorithm.
- Explain briefly why your algorithm is $O(|E|\log|E|)$.

Login: ______________________________

Problem 4
Consider the problem of finding the length of the *longest* path from a vertex s to a vertex t in a *directed acyclic graph* $G = (V, E)$ with edge-weights w_e.

- *Student A* thinks that they can solve this problem using an altered Dijkstra's Algorithm that stores the maximum cost of each node and uses a maximum cost priority queue (a priority queue that returns the maximum value instead of the minimum value). Why would this *not* work?
- Describe and write pseudocode for an algorithm that would work. If there exists no path between s and t, then return ∞.

Login: ________________________

Problem 5
Consider the following two algorithms, each of which takes an input graph, G = (V, E) with edge lengths l(u, v).

Algorithm A: This algorithm picks a random node s, runs Dijsktras algorithm on it to find distances from s to all nodes, and retains the $|V|/2$ nodes that are closest to s [in the case of a tie, it chooses randomly.]. It then recursively calls itself on the graph consisting of these nearby nodes and all edges in E that join such nearby nodes. For a graph with a single vertex, the algorithm returns that node. This particular implementation of Dijkstra uses an array to represent the priority queue.

Algorithm B: The second algorithm sorts the edges by length, and returns the length of the median edge.

What are the runtimes of each algorithm? For what kinds of graphs would Algorithm A be faster? Algorithm B? Briefly explain your answers.

Problem 6
Given an input string and a dictionary of words, develop an efficient algorithm to determine if the input string can be segmented into a space-separated sequence of dictionary words. There is no limit to the number of times you may use each word in the dictionary.

For example, suppose we have the following dictionary of known words:

```
dictionary = {"i", "like", "ice", "cream", "icecream", "popsicle"}
```

Consider the following input/output pairs.

```
Input: "ilike"
Output: True
We can segment the string into "i like" using words from the dictionary.

Input: "ilikeicecream"
Output: True
We can represent the string as either "i like icecream" or "i like ice cream."

Input: "ieatpopsicles"
Output: False
We can’t segment the string into words contained in our dictionary.
```

Note that a greedy approach will not always guarantee the correct solution! For example, consider the following test case:

```
dictionary = {"a", "an", "ill", "pill", "pillow", "case"}
Input: "pillowcase"
Output: True
We can segment the string into "pillow case."
```

In this case, if we tried to find the first matching prefix, we would pick “pill.” However, that makes the rest of the string (“owcase”) unsegmentable given our dictionary.

Write pseudocode for your algorithm.

Login: ____________________________

Problem 7
Write pseudocode for a method $reverseList(head)$ that takes in the head of a singly linked-list and 'reverses' the list. Note that a singly linked-list is simply a linked list in which every node has a pointer to its next element in the list and the last node's next pointer points to $null$. Example: Given a linked list$A \rightarrow B \rightarrow C \rightarrow D$ with head A the method turns it into the list: $D \rightarrow C \rightarrow B \rightarrow A$ with head D. Explain the runtime of your solution.

Problem 8
You are given a non-empty stack S of integers. Write pseudocode for a method that sorts the stack in ascending order from the bottom-up (i.e. the bottom element is the smallest integer). You are allowed to use the *push*() , *pop*(), *peek*(), *isEmpty*() methods of a stack. You may use another stack in order to solve this problem, but you may not use other data structures in your solution.
Hint : You can accomplish this by creating one more stack in your method, and transfer elements into the new stack each time checking that the transfer doesn't violate the sort order.
Example : Here's a stack S
— 5 —
— 2 —
— 3 —
— 1 —
— 4 —

Your method should return a stack that looks like this:
— 5 —
— 4 —
— 3 —
— 2 —
— 1 —

Login: ____________________

Problem 9

1. What is the purpose of the learning rate η?

2. Name two defining aspects of the functional paradigm and describe them in a few sentences.

3. Are there graphs where Dijkstra's will fail? If so, explain why. If not, explain why not.

4. Describe one difference between problems in the sets P and NP.

5. What is the difference between online algorithms and offline algorithms? Give an example of a time that we have used each in class.