Name:

Penn State access ID (xyz1234) in the following box:

Instructions:

- All questions are weighted equally.
- Please clearly write your name and your PSU access ID (i.e., xyz1234) in the box on top of **every page**.
- Write your solutions only in the space provided. There is one answer sheet for each problem. **Do not write outside of the black bounding box**: the scanner will not be able to read anything outside of this box.
- We are providing one extra page at the end if you need extra space. Make sure you mention in the space provided that you answer continues there.
- **Important:** Brief but precise description of algorithms is required (try less than 6 sentences if you can, but don't fret if you go over). Pseudocode is not required, but you may provide it if you think it helps your exposition. Pseudocode alone will not receive any credits. Do not forget to provide the running time analysis of all your algorithms.

You have three hours. Good luck!

1. Induction

Prove the following statements using mathematical induction.

(a) If $S(n) = 2 \cdot S(n-1) + 1$ and $S(0) = 1$, then for all $n \geq 0$, $S(n) = 2^{n+1} - 1$.

(b) For all $n \geq 0$, $\sum_{i=0}^{n} i^2 = O(n^3)$.

(c) For all $n \geq 1$, $\sum_{i=1}^{n} \frac{1}{2^i} = 1 - \frac{1}{2^n}$.

2. Rates of growth

List the following six functions so that they are in order from smallest to largest when n is very large. Indicate which functions are Θ of each other.

$$\begin{aligned} f_1(n) &= (\log n)^{\log n} \\ f_2(n) &= \log(n^n) \\ f_3(n) &= (n/\log n)^{\log\log n} \\ f_4(n) &= n^3 \cdot n^{\log\log\log n} \\ f_5(n) &= n \cdot 2^{\sqrt{\log n}} \end{aligned}$$

Note: To receive partial credit for a not entirely correct answer, you must show your work neatly, and your work should not simply consist of evaluating the functions at a large value of n in the hope that this value is large enough to reveal the correct answer.

3. Recurrences

Solve each of the following recurrences. Giving your solution in O-notation suffices. You may assume that n is of some special form (e.g., a power of two or of some other number), and that the recurrence has a convenient base case that is $\Theta(1)$.

(a) $T(n) = 2T(n/4) + 10n$.

(b) $T(n) = T(n-1) + 2^n$.

(c) $T(n) = 32T(n/16) + n^{3/2}\log n$.

4. Shortest path

Let $G = (V, E)$ be an undirected weighted graph, where the weight of each edge in an integer from the set $\{1, 2, 3, 4, 5\}$. Provide an algorithm that given as input two vertices u and v of G, returns the weight of the shortest path between u and v. (The weight of a path is the sum of the weights of the edges in the path; the shortest path between two vertices is the path of minimum weight between the vertices.) Prove the correctness and the running time of your algorithm. To receive full credit, the running time of your algorithm should be $O(|V| + |E|)$ and you may assume that G is given as an adjacency list.

Hint: "Off-the-shelf" shortest path algorithms such as Dijkstra's of Bellman-Ford do not have the required $O(|V| + |E|)$ running time.

5. Divide and Conquer

Given k sorted lists of length n, give an algorithm to produce a single sorted list containing the elements of all the lists. Prove the correctness and the running time of your algorithm. To receive full credit, the running time of your algorithm should be $O(kn\log k)$.

6. Counting Circuits and Functions

Recall that a Boolean circuit with k gates and n Boolean inputs is a directed acyclic graph with n input nodes and k additional nodes called "gates" where each gate (node) has at most 2 inputs (fan-in 2), which can be either other gate or an input. Each gate computes either OR, AND, NOT of its inputs depending on its specified type. Finally, there is an output gate: a specified gate which will be the output of the circuit. For example, with two Boolean inputs, one can design a simple circuit that outputs AND of the inputs by creating an AND gate with two inputs as its input, and specifying it as the output gate.

(a) Show an upper bound on the total number of possible Boolean circuits (with fan-in 2, OR, AND, NOT gates) computed by k many gates. (Hint: each gate can be described by its gate type and its at most two input wires. What is the number of possibilities for each gate? If there are k many gates, what is the total number of possibilities?)

(b) Count the total number of possible Boolean functions $f : \{0,1\}^n \to \{0,1\}$ with n inputs.

(c) At what number k is the bound given at (a) lower than the number given at (b)?

7. Dynamic Programming

Let $S = (s_1, \ldots, s_n)$ be a sequence of n distinct integers. A subsequence $(s_{i_1}, \ldots, s_{i_k})$ of S (with $1 \le i_1 < i_2 < \cdots < i_k \le n$) is called a *zig-zag* subsequence if either:

$$s_{i_1} < s_{i_2} \text{ and } s_{i_2} > s_{i_3} \text{ and } s_{i_3} < s_{i_4} \ldots,$$

or

$$s_{i_1} > s_{i_2} \text{ and } s_{i_2} < s_{i_3} \text{ and } s_{i_3} > s_{i_4} \ldots$$

For example, if $S = (1,3,7,2,3,6,9,-2,3)$, then $(1,7,2,9,-2,3)$, $(7,9,-2)$ and $(7,2,9)$ are all zig-zag subsequences of S and $(1,3,7)$, $(2,6,9)$ are not.

Design an algorithm that computes the length of the longest zig-zag subsequence of a given sequence S. For full credit, the running time of your algorithm should be $O(n^2)$.

Name:	**PSU Access ID (xyz1234):**

Problem 1.

Name:	**PSU Access ID (xyz1234):**

Problem 2.

Name:	**PSU Access ID (xyz1234):**

Problem 3.

Name:	**PSU Access ID (xyz1234):**

Problem 4.

Name:	**PSU Access ID (xyz1234):**

Problem 5.

Name:	**PSU Access ID (xyz1234):**

Problem 6.

Name:	**PSU Access ID (xyz1234):**

Problem 7.

Name:	**PSU Access ID (xyz1234):**

Extra.