• This is a closed-book, closed-notes, open-brain exam. If you brought anything with you besides writing instruments and your handwritten $8\frac{1}{2}'' \times 11''$ cheat sheet, please leave it at the front of the classroom.

• Print your name, netid, and alias in the boxes above. Print your name at the top of every page (in case the staple falls out!).

• You should answer all the questions on the exam.

• The last few pages of this booklet are blank; use them for scratch paper. Please let us know if you need more paper.

• If your cheat sheet if not handwritten by yourself, or it is photocopied, please do not use it and leave it in front of the classroom.

• Submit your cheat sheet together with your exam. An exam without your cheat sheet attached to it will not be checked.

• If you are NOT using a cheat sheet you should indicate it in large friendly letters on this page.

• There are 8 multiple-choice questions and 2 word problems. For each of the word problems, you must give and prove the desired result. For each of the multiple-choice questions, there is only a single correct answer. A correct answer will gain you five points, whereas an incorrect answer will gain you zero points (i.e., no penalty for incorrect answers). In the case of several “correct” answers, you should choose the one which is best.

• Only for questions 9 and 10, answers containing only the expression: “I don't know”, will get 25% of the points of the question. If you write anything else, it would be ignored. There is no “I dont know” option for multiple choice questions.

• Time limit: 75 minutes.

• Relax. Breathe. This is just an easy, silly and stupid midterm.

<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 8</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
1. **[5 Points]** The solution to the recurrence $H(n) = n + \sqrt{n} \cdot H(\lfloor \sqrt{n} \rfloor)$ is

A. $\Theta(n)$.
B. $\Theta(\sqrt{n})$.
C. $\Theta(n \log \log n)$.
D. $\Theta(\log n)$.
E. $\Theta(\log^* n)$.
F. None of the above.

**Answer:** 

2. **[5 Points]** Given two graphs $G_1$ and $G_2$, deciding if one can delete $k$ edges from $G_1$ and get the graph $G_2$ (we consider two graphs to be the same if one can rename the vertices of one graph to get the second graph) is

A. $NP$-complete.
B. Solvable in polynomial time.
C. None of the above.

**Answer:** 

3. **[5 Points]** Statement: “Any problem that can be solved by an algorithm that uses only $O(\log n)$ bits of memory (in addition to the input, which resides in a read-only memory) can be solved using polynomial time.” This statement is:

A. False.
B. True.
C. False only if $P = NP$.
D. True only if $P = NP$.
E. None of the above.

**Answer:** 

4. **[5 Points]** Consider the following functions:

- $\lg(\lg n)$, $(1 + \frac{1}{1000})^n$
- $2^n$, $(\lg n)^{\lg^* n}$, $(\lg^* n)^{\lg n}$, $(\lg n)^{\lg n}$.

What is the correct ordering of their asymptotic growth rates?
A. \( \lg(lg n) \ll (lg^* n)^{lg n} \ll (lg n)^{lg n} \ll (1 + \frac{1}{1000})^n \ll 2^n \)

B. \((lg^* n)^{lg n} \ll \lg(lg n) \ll (lg n)^{lg n} \ll (1 + \frac{1}{1000})^n \ll (lg n)^{lg^* n} \)

C. \((lg^* n)^{lg n} \ll (1 + \frac{1}{1000})^n \ll (lg n)^{lg^* n} \ll 2^n \ll (lg n)^{lg n} \ll (lg n) \)

D. \(\lg(lg n) \ll (lg^* n)^{lg n} \ll (1 + \frac{1}{1000})^n \ll 2^n \ll (lg n)^{lg^* n} \ll (lg n)^{lg n} \)

E. None of the above

Answer: 

5. [5 Points] The problem **Triple 2Coloring** (deciding if the vertices of a graph \( G \) can be partitioned into three sets \( S, T, V \), such that the induced subgraphs \( G_S, G_T, G_V \) are each colorable by two colors) is:

A. NP-Complete.
B. Solvable in polynomial time.
C. NP-Hard.
D. None of the above.

Answer: 

6. [5 Points] Given a graph \( G \) with \( n \) vertices, deciding if there is a clique of size \( \geq 195 \) is:

A. Solvable in polynomial time.
B. NP-Complete.

Answer: 

7. [5 Points] The solution to the recurrence

\[ A(n) = A(\lfloor \log n \rfloor) + 1 \]

is:

A. \( \Theta(1) \)
B. \( \Theta(\log \log \log n) \)
C. \( \Theta(n \log n) \).
D. \( \Theta(n \log n) \).
E. \( \Theta(\log \log n) \).
F. \( \Theta(\log n) \).
8. **[5 Points]** Given a boolean formula $F$ of length $n$ defined over 100 variables, deciding if $F$ is satisfiable can be done in:

A. $O(2^n)$ time, and there is no faster algorithm.
B. $O(\log \log n)$ time.
C. Polynomial time.
D. This is an NP-complete problem, and it cannot be solved.
E. Constant time.
F. None of the above.
G. All of the above.

**Answer:** 

9. **THE SATISFIABILITY FAUN**

**[30 Points]**

The satisfiability faun, the uncle of the Partition satyr and the Deduction fairy, came to visit you during the holidays, and gave you as a token of appropriation a black box $B$ that can solve the 3SAT decision problem in constant time. Namely, given any 3SAT formula $F$, the procedure $B(F)$ returns true if $F$ is satisfiable, and false otherwise. Show an algorithm that, given a 3SAT formula $F$, finds a satisfying assignment for $F$, if such an assignment exists, using the black box. How fast is your algorithm? (Recall that a 3SAT formula must have exactly three literals in each clause.)

Write short and concise solution — you would lose points for a redundantly long answer.

10. **GENERATING STRINGS.**

**[30 Points]**

You have a table of rules, where the $i$th rule looks like $a_i \rightarrow b_ic_i$, where $a_i$, $b_i$, $c_i$ are all characters taken from a finite set $\Sigma$ of possible characters, where $\Sigma$ has a constant number of elements. Similarly, the table of rules contains a constant number of such rules.

A string is generated as follows. You start from a string containing a single character “a”. Next, at each stage, you can take one character $c$ in the string, and replace it by two characters $vw$, assuming that the rule $c \rightarrow vw$ appears somewhere in the table of rules.
Given a string $A = a_1a_2\ldots a_n$, show an algorithm for deciding if this string can be generated by this set of rules. How fast is your algorithm?

For example, if the set of rules is

- $a \rightarrow bc$
- $a \rightarrow cd$
- $b \rightarrow ca$
- $b \rightarrow dd$,

then one can generate the string “cddce” as follows: “$a$” $\rightarrow$ “$bc$” $\rightarrow$ “$cac$” $\rightarrow$ “$cbcc$” $\rightarrow$ “$cddce$”.

Write short and concise solution — you would lose points for a redundantly long answer.