1	2	3	4	Σ
10	30	30	30	100

Student number	2	0						
Name								

Final Exam — Introduction to Algorithms (CS 300A)

June 11, 2018, 13:00–15:40

Instructions:

- Before you start: **Write your name and student number** (one digit per square!) **clearly** on **all three pages** of this exam booklet.
- This booklet has **three pages** with **four problems** in total. Check that you have all!
- The space provided is sufficient for your answer. If you need scratch space, use the back side. If you need more space for your answer, you can also use the back side (but *indicate clearly on the front side* that your answer continues on the back).
- This is a **closed book exam**. You are not allowed to consult any book or notes.
- Your answers must be in **English**. Write clearly!
- To ensure a quiet exam environment, we will **not answer questions** during the exam. If you think there is a mistake in the question, explain so, and use common sense to answer the question.
- **Relax. Breathe.** This is just an easy, silly, and stupid final exam.

Problem 1: (10 pts) State the *Non-Polynomial Time Hypothesis.*

Problem 2: (30 pts) You are studying a fascinating and difficult problem called FinalExam. Your friend, who is really smart, has proven three polynomial-time reductions involving this problem:

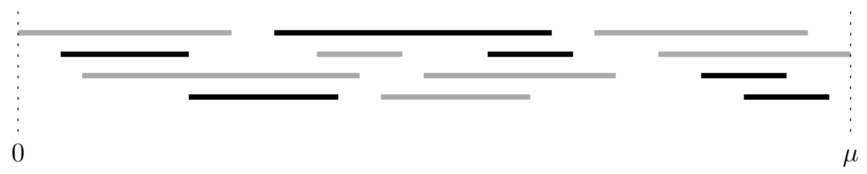
$$\textsc{FinalExam} \longrightarrow \textsc{HamiltonianCycle},$$
$$\textsc{Clique} \longrightarrow \textsc{FinalExam},$$
$$\textsc{EditDistance} \longrightarrow \textsc{FinalExam}.$$

Which of these three reductions is the most interesting? What does it imply, and why? What do the other two reductions imply?

Student number	2	0						
Name								

Problem 3: (30 pts) Let X be a set of n intervals on the real line, whose left and right endpoints are given by the arrays `L[1..n]` and `R[1..n]`. Together, the intervals cover the interval $[0, \mu]$, for some $\mu > 0$, and all endpoints are distinct (that is, all the $2n$ coordinates in the two arrays `L` and `R` are different numbers).

A subset $Y \subseteq X$ is a *cover* if the intervals in Y *also* cover the entire interval $[0, \mu]$. In the figure below, the grey intervals form a cover of size seven.



The following greedy algorithm computes a cover for X of the smallest possible size:

```
SmallestCover(X):
  Let p be the unique interval with L[p] == 0
  Let Y = { p }
  repeat:
    remove from X all intervals i with R[i] <= R[p] (including p itself)
    if X is empty:
      return Y
    let Z be all intervals i in X with L[i] < R[p]
    let q be the interval in Z that maximizes R[q]
    Y := Y ∪ { q }
    p := q
```

(a) (10 pts) Explain why the output of this algorithm is indeed a cover for X. *(Hint: Use a loop invariant.)*

(b) (20 pts) Now prove that the output is indeed a smallest cover for X.

You will receive 5 points if you write "I don't know" and nothing else.

Student number	2	0						
Name								

Problem 4: (30 pts) A string x is a *supersequence* of a string y if we can obtain x by inserting zero or more letters into y, or equivalently, if y is a subsequence of x. For example, the string `DYNAMICPROGRAMMING` is a supersequence of the string `DAMPRAG`.

A *palindrome* is any string that is exactly the same as its reversal, like `I`, `DAD`, `HANNAH`, `AIBOHPHOBIA` (fear of palindromes), or the empty string.

We want to use dynamic programming to find the length of the shortest supersequence x of a given string y such that x is also a palindrome. For example, the 11-letter string `EHECADACEHE` is the shortest palindrome supersequence of `HEADACHE`, so given the string `HEADACHE` as input, the algorithm should output the number 11.

We denote the input string as `Y[1..n]`.

(a) (5 pts) Describe the subproblems appearing in *ShortestPalindromicSupersequence*:

`SPS[          ] =`

(b) (15 pts) Give the recursion formula (do not write pseudo-code!) that finds the solution to a subproblem based on the solution to smaller subproblems. Explain and justify your answer!

(c) (3 pts) What is a good data structure to store the solutions to the subproblems?

(d) (4 pts) In what order should we solve the subproblems? (Now you can write pseudo-code, if you want.)

(e) (3 pts) What is the running time of the algorithm? (No explanation needed.)