NAME (please PRINT in large letters):

SECTION: 01 02 (circle one)

CMSC 27200 Theory of Algorithms
Second Midterm — 02-26-2015

The exam is **closed book.** Do not use notes.
The use of ELECTRONIC DEVICES is strictly forbidden.

Use the paper provided. Do not use your own paper. You may continue on the reverse side of each sheet.

SHOW ALL YOUR WORK. Correct conclusions that come out of the blue (details of calculation missing) will not receive credit.

Please be accurate and concise. When writing **pseudocode,** keep with simple instructions and notation, do not use notation specific to some programming languages. **Define your variables** and comment on each line, what is happening.
You do NOT need to prove **correctness** of your algorithms unless specifically asked.

Warning: The **extra-credit (XC)** problems are underrated; do the ordinary problems before you attempt the extra-credit problems.

This exam contributes 20% to your course grade.
The total point value of the ordinary (non-extra-credit) problems is 200 points. The extra-credit problems, as their name suggests, add to your score.

1. **(1 point)** Spell the singular of the word "vertices."

2. (a) **(6 points)** Prove: the function $2^{\sqrt{n}}$ is not polynomially bounded in n, i. e., there is no constant C such that $2^{\sqrt{n}} = O(n^C)$.

   (b) **(12 points)** Recall the Integer Knapsack problem:

   INPUT: a list of n positive integers $w_1, \dots, w_n$ (the weights);
   a list of n positive integers $v_1, \dots, v_n$ (the values);
   a positive integer W (the weight limit).
   GOAL: maximize the value $\sum_{i \in I} v_i$ over all subsets
   $I \subseteq \{1, \dots, n\}$ satisfying the constraint $\sum_{i \in I} w_i \le W$.
   OUTPUT: this maximum value.

   We solved this problem in $O(nW)$ steps using dynamic programming. Prove: this is NOT a polynomial-time algorithm. – Explanation: You need to construct an infinite set of instances where the algorithm takes more than polynomial time.

3. (5+5+5+5+5+3+3 points) (polynomial time) For each function below, decide whether or not it is computable in polynomial time. The input is the number n in binary.

   Clearly state your answers: YES or NO. Prove your "NO" answers to the extent current knowledge permits. If your answer is YES, briefly sketch an algorithm (either in clear English or in pseudocode or a combination of these). Prove that it runs in polynomial time. Use algorithmic principles studied in class; do not use Newton's method. The goal is not to state the most efficient algorithm but the simplest algorithm that justifies the polinomial time claim. Prove that your algorithm runs in polynomial time.

   (a) $n!$ (b) $\binom{n}{5}$ (c) $\binom{2n}{n}$ (d) $n^{\lfloor \lg n \rfloor}$ (e) $\lfloor \sqrt{n} \rfloor$ (f) the smallest prime factor of n (g) $\pi(n)$ (the number of primes $\leq n$)

4. (a) **(XC, 5 points)** Let B be a Bernoulli trial with probability p of success. (So X is a random variable that takes values 0 and 1 with $P(B = 1) = p$.) Repeat B until the first success; let X denote the number of times B was performed. Prove: $E(X) = 1/p$.

   (b) **(10 points)** Alice needs to find a random n-digit prime. (Digits are in binary; initial zeros are permitted.) She keeps trying random n-digit numbers until she finds one that is prime. (She has a primality tester.) Let X denote the number of times she tries. Asymptotically evaluate $E(X)$ (as a function of n). Your answer should be of the form $E(X) \sim an^bc^n$; determine the constants a, b, c. Justify your answer. You may use part (a) without proof.

5. (15 points) (Min-cost connected spanning subgraph)
Let $G = (V, E, w)$ be a connected undirected weighted graph where $w : E \to \mathbb{R}$ is the weight function. (Negative weights are permitted.) Our goal is to find a subset $F \subseteq E$ of the edges such that the subgraph (V, F) is connected; and minimize the total weight of such a subset. Design an algorithm that finds such a subset in "Dijkstra time" $O(|V| \log |V| + |E|)$ (the optimal time bound for Dijkstra's algorithm). Describe your algorithm in unambiguous, concise English, no pseudocode needed. Briefly justify (a) optimality and (b) the time bound.

6. (25 points) (Dijkstra with a twist) We are given a weighted digraph with a source vertex, $G = (V, E, s, w)$ where $s \in V$ is the source and $w : E \to \mathbb{R}$ is the weight function. All weights are non-negative. We are also given a partition of the edges into "red" and "blue" edges: $E = R \cup B$, where $R \cap B = \emptyset$. For all vertices $v \in V$ find the minimum cost of reaching v from s along a path that uses at most one red edge. Your algorithm should run in "Dijkstra time." – Instruction: do not invent a new algorithm; apply an algorithm studied in class to an input other than G. Your job is to construct the new input. Describe your construction in unambiguous English and mathematical formulas. No pseudocode needed.

7. (5+5+5 points) (The complexity class P) For each problem below decide whether or not is belongs to the complexity class P. Clearly state your answer (YES or NO). Reason your answers. In the case of "YES" answers, sketch (in English) and name the algorithm involved in the justification of your answer. Make an accurate statement of the running time and justify it. All integers are given in binary.

   (a) compute the gcd of a pair of integers

   (b) given a weighted digraph with integer weights, decide whether or not there is a negative cycle

   (c) given the integers a and m, decide whether or not a has a multiplicative inverse modulo m.

8. (5+5+3+8+10 points) (Jarník vs. Dijkstra) Recall that Jarník's (a.k.a. Prim's) algorithm solves the min-cost spanning tree problem.

   (a) State exactly the input that Jarník's algorithm takes; compare it item by item with the input taken by Dijkstra's algorithm. Pay special attention to whether the graph is directed or undirected, whether the weights need to be non-negative; whether the input specifies a source node; connectivity issues.

   (b) Describe the output of (b1) Dijkstra's (b2) Jarník's algorithm

   (c) State the three data structure operations used, and how many times each was used (upper bound) in each algorithm

   (d) Recall the last and decisive relative loop invariant for Dijkstra: "for each $v \in V$, the quantity $c(v)$ is the min cost of all $s \to \cdots \to v$ paths that pass through black vertices only (except for v itself)." State the analogous loop invariant for Jarník. Do not prove.

   (e) Describe Jarník's algorithm in pseudocode. Define your variables. Indicate where the code differs from Dijkstra's. (Use the reverse side of this sheet.)

9. (8+3+3+3+3 points) (loop invariants)

(a) Give an exact definition of a loop invariant. Include the definition of the configuration space. Make sure to state the domain of your predicates and the domain and range of your functions.

(b) The body of Dijkstra's algorithm is a single **while** loop. For each statement below, decide whether or not it is a loop invariant. Briefly justify your answers. (The vertices are denoted $v_1, \dots, v_n$.)

(b1) Vertex v_7 is black.

(b2) Vertex v_7 is white.

(b3) $c(v_7) \leq 5.95$. (c denotes the current cost.)

(b4) The quantity $c(v_7)$ cannot increase.

10. (5 points) Spell out the acronym "NP."

11. (15 points) All graphs in this problem are undirected. Recall that a *legal coloring* of a graph an assignment of colors to the vertices such that adjacent vertices always get different colors. We say that a graph is k-*colorable* if it has a legal coloring with $\leq k$ colors.

    Let k-COL denote the decision problem that takes an undirected graph as input and decides whether or not it is k-colorable.

    Give a Karp-reduction from 3-COL to 4-COL. – Explanation. Given a graph G, you need to construct in polynomial time a graph $f(G)$ such that $f(G)$ is 4-colorable if and only if G is 3-colorable. Explain yor algorithm in English, no pseudocode required. You need to justify the "if and only if" statement.

12. (14 points) A *palindromic string* is a string that reads the same forward and backward; for instance ABBA, ACA, AAAAA are palindromic strings but ABAB is not.

    By a *substring* we mean a *contiguous* substring; for instance, given the string ABCDE, the strings ABC and BCDE are substrings but ACDE is not.

    Given a string of length n over the English alphabet, find the length of the longest palindromic substring. (Example: the word GATTACA includes the following palindromic substrings of length ≥ 2: TT, ACA, ATTA. So the answer will be "4.") Use dynamic programming in $O(n^2)$ steps. Half the credit goes for a clear definition of the variables you introduce (the "brain" of the algorithm, not to be confused with the recurrence, the "heart" of the algorithm). Pseudocode required.

13. (XC, 20 points) Let $G = (V, E, w)$ be an undirected weighted graph and let $A \subset V$ be a nonempty proper subset of V (so $A \neq V$). The *separation* $\mathrm{sep}(A)$ is defined as

$$\mathrm{sep}(A) = \min\{w(a, b) \mid a \in A, b \in V \setminus A\}.$$

Given G, find a subset A with maximum possible separation. Do this in linear time. Explain your algorithm in clear English. No pseudocode needed. Justify the running time.