Name ______________________________

Student Number ______________________________

HAND IN
answers recorded
on question paper

BISHOP'S UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

CS 467/567

SAMPLE FIRST EXAMINATION

29 February 1901

Instructor: Stefan D. Bruda

Instructions

- This examination is *80 minutes in length* and is *open book*. You are allowed to use any kind of printed documentation. Electronic devices are not permitted. You are *not* allowed to share material with your colleagues. *Any violation of these rules will result in the complete forfeiture of the examination.*
- There is no accident that the total number of marks add up to the length of the test in minutes. The number of marks awarded for each question should give you an estimate on how much time you are supposed to spend answering the question.
- *To obtain full marks provide all the pertinent details*. This being said, do not give unnecessarily long answers. In principle, all your answers should fit in the space provided for this purpose. If you need more space, use the back of the pages or attach extra sheets of paper. However, if your answer is not (completely) contained in the respective space, clearly mention within this space where I can find it.
- Make sure that your name and student number appear on top of each sheet which is not securely stapled to the booklet (just in case). This also applies to any sheet which you detach from the booklet.
- The number of marks for each question appears in square brackets right after the question number. If a question has sub-questions, then the number of marks for each sub-question is also provided.

When you are instructed to do so, turn the page to begin the test.

1			/	10			
2			/	10			
3	a,b,c		/	30			
4	a,b		/	15			
5			/	10			
6			/	5			
	Total:		/	80	=	/	20

1. [10] Consider a domain D of *individuals*. A *predicate* is a named relation over D^k for some $k \geq 0$. A *literal* is either a predicate instance whose arguments are either individuals or *variables*, or the negation of a predicate. A *clause* is a disjunction of literals. A *first-order logic formula* is a conjunction of clauses. Here is an example of first-order logic formula, where variables are capitalized and individuals are written in lower caps:

$$(\overline{\text{dog}(fido)} \vee \text{dog}(X) \vee \text{smelly}(fido)) \wedge (\overline{\text{dog}(fido)} \vee \text{smelly}(X) \vee \text{smelly}(fido))$$

An *interpretation* maps each variable X to a value in D and each predicate $p(x_1, \ldots, x_k)$ to either True or False.

The *first-order satisfiability problem* (FOL-SAT) is stated as follows: Given a first-order logic formula, determine whether there exists an interpretation that makes the formula true (case in which the formula is deemed satisfiable).

Find a polynomial reduction from SAT to FOL-SAT. Justify your answer formally.

ANSWER:

---

2. [10] Let $\mathbb{A}$ and $\mathbb{B}$ be two problems. Let τ be a reduction from $\mathbb{A}$ to $\mathbb{B}$ (meaning that $w \in \mathbb{A}$ iff $\tau(w) \in \mathbb{B}$), where $\tau(w)$ for some string w of length n is computed as follows:

   (a) Find two numbers p and q such that $1 < p < q < n$ and $p \times q = n$
   (b) If such numbers exist then return the first p symbols of w followed by the last q symbols of w
   (c) If such numbers do not exist then return w unchanged

   Suppose that $\mathbb{A}$ is NP-complete. What can you say about $\mathbb{B}$? Explain.

ANSWER:

---

3. [30] The SCHEDULING WITH DEPENDENCIES problem is stated as follows: *We are given a set of jobs $J = \{1, 2, \ldots, n\}$ with running times $t_i \geq 0$, $1 \leq i \leq n$, respectively. We are also given a directed acyclic dependency graph $G = (J, E)$ such that $(i, j) \in E$ iff job i must be completed before job j starts. Given $m > 0$, find a schedule for all the jobs in J on m identical machines such that the overall running time is minimized.*

   Consider the following candidate for an approximation algorithm for SCHEDULING WITH DEPENDENCIES, called *list scheduling*: Whenever a machine is idle, we schedule the first available job (with no predecessors in G) and then we delete from J the vertex j and from E all the edges (j, u). We repeat this until no more jobs are available.

   (a) [5] Show that list scheduling runs in polynomial time.

   ANSWER:

   (b) [5] Let $p = (p_1, p_2, \ldots, p_k)$ be a path in the graph G. Define the running time of p as $T_p = \sum_{i=1}^{k} t_{p_k}$ (the sum of the running times of the jobs along the path). Furthermore, let T^* be the running time of the optimal schedule.
   Prove that T^* is larger than T_p for any path p in G.

   ANSWER:

(c) [20] Show that list scheduling is a 2-approximation scheme for SCHEDULING WITH DEADLINES

In doing so, let x be the job that completes last and let p_x be a path that starts with a job with no predecessors and end with x. At any moment in time the machines either execute a job from p_x or they do not. They are never idle when not executing a job from p_x however, for if they were then whatever job from p_x is available will be executed. Split the time units into two sets A and B, where A are the time units in which a job from p_x executes and B the other time units. Establish suitable upper bounds for both $|A|$ and $|B|$ and you should be done.

ANSWER:

---

4. [15] Suppose we want to formulate HAMILTONIAN CYCLE as a linear programming problem. Note that this is a decision rather than an optimization problem.

   (a) [5] What is the objective function for such a formulation? Explain.

   ANSWER:

   (b) [10] Define the following property of Hamiltonian cycles as linear constraints: Given a graph $G = (V, E)$, any vertex $v \in V$ has exactly 2 incident edges belonging to the Hamiltonian cycle of G.

   Hint. You may want to use the variables b_e, $e \in E$ such that $b_e = 1$ if e belongs to the Hamiltonian cycle and $b_e = 0$ otherwise.

   ANSWER:

   ---

5. [10] Formulate the following problem as a linear programming problem:

   Given a graph $G = (V, E)$ and two designated vertices $s, t \in V$, find the maximum number of edge-disjoint paths from s to t (that is, no two paths have a common edge common edges).

   Hint. Can you formulate this problem as a maximum flow instance?

ANSWER:

---

6. [5] The polar angle of a point p is the angle between the segment $\overrightarrow{(0,0)p}$ and the x axis. Explain how to use the cross product to compare the polar angles of two points p_1 and p_2.

ANSWER:
