Name ______________________ Student No. ______________________

No aids allowed. Answer all questions on test paper. Use backs of sheets for scratch work.

Total Marks: 50

1. Let $G = (V, E)$ be a flow network, with source s and sink t, and with the capacity function $c(u, v) \geq 0$. A flow in F is a real-valued function $f : V \times V \longrightarrow \mathbb{R}$ that satisfies the following three properties; define them in the space provided:

[2] Capacity constraint: For all $u, v \in V$, $f(u, v) \leq c(u, v)$.

[2] Skew symmetry: For all $u, v \in V$, $f(u, v) = -f(v, u)$.

[2] Flow conservation: for all $u \in V - \{s, t\}$, $\sum_{v \in V} f(u, v) = 0$.

[2] Define what it means for a problem P_1 to be polytime reducible to a problem P_2 (denoted $P_1 \xrightarrow{p} P_2$):

$P_1 \xrightarrow{p} P_2$ if there is a polytime algorithm for P_1 which is allowed to access a solver for P_2. The time taken by the P_2 solver is not counted.

[2] Define what it means for a language L_1 to be polytime-many-one reducible to a language L_2 (denoted $L_1 \leq_p L_2$):

$L_1 \leq_p L_2$ if there is a polytime function f such that $x \in L_1 \iff f(x) \in L_2$.

[2] State the Polytime Thesis:

A decision problem D can be solved by a polytime algorithm iff L_D can be recognized by some polytime TM M, i.e. $L_D = L(M)$.

[2] Give the informal definition of NP:

$L \in$ NP iff L has a polytime "guess-and-check" algorithm.

[2] and show how to use this definition to prove that SKD (Simple Knapsack Decision) is in NP:

On input $I = \langle d, w_1, \ldots, w_d, C, B\rangle$, *guess* a subset $S_0 \subseteq \{1, \ldots, d\}$ and *check* that $B \leq \sum_{i \in S_0} \leq C$.

[2] Now give the formal definition of NP: let $\Sigma = \{0, 1\}$. Then, a language $L \subseteq \Sigma^*$ is in NP iff:

there exists a relation $R(x, y)$, $x, y \in \Sigma^*$, which is recognized by some two-input TM M which runs in time polynomial in $|x|$, such that $x \in L \iff \exists y R(x, y)$ holds.

[2] and explain very briefly how the formal definition of NP is related to the informal definition of NP:

guess y_0 and *check* that $R(x, y_0)$ holds.

[2] and show how to use the formal definition to show that SKD is in NP:

Let $x = \langle d, w_1, \ldots, w_d, C, B\rangle$ and let $y = \langle S_0\rangle$, and let $R(x, y)$ be $B \leq \sum_{i \in S_0} w_i \leq C$.

2. We define a circuit C with n input gates as follows:

$$C = x_1, \ldots, x_n, x_{n+1}, \ldots, x_m$$

where the first n variables are the inputs, and for $n+1 \leq i \leq m$, each x_i is computed by one of the following three rules:

$$x_i \longleftarrow x_j \vee x_k$$
$$x_i \longleftarrow x_j \wedge x_k$$
$$x_i \longleftarrow \neg x_j$$

where $1 \leq j, k < i$. The output of C is x_m.

[2] Define the problem CircuitSat:

Input: $\langle C \rangle$ where C is a circuit on n inputs.

Question: Is there some input $a_1 \ldots a_n \in \{0,1\}^n$ such that the output of C on this input is 1?

[2] Now define what it means for a propositional formula to be in 3CNF:

A clause is a disjunction of literals, i.e. $L_1 \vee \ldots \vee L_n$. A formula is in CNF if it is a conjunction of clauses, i.e. $C_1 \wedge \ldots \wedge C_m$. A formula is in 3CNF if each clause has at most 3 literals.

[2] and define the problem 3SAT:

Input: $\langle F \rangle$ where F is a formula in 3CNF

Question: Is F satisfiable?

As was stated in class, CircuitSat is an NP-complete problem. Prove that 3SAT is an NP-complete problem.

[2] First show briefly that 3SAT is in NP:

guess a truth value assignment t to F, and *check* that t indeed satisfies F. Since 3SAT has a polytime "guess-and-check" algorithm, it follows that 3SAT is in NP.

Now show that 3SAT is NP-hard by following the outline below.

[2] State clearly what are you reducing to what: CircuitSat $\leq_p$ 3SAT.

[10] Give the reduction (you do not have to prove correctness):

We want a function f such that $f(C) = F_C$, and C is a satisfiable circuit iff F_C is a satisfiable formula.

Given $C = x_1, \ldots, x_n, x_{n+1}, \ldots, x_m$, convert x_i, for $n+1 \leq i \leq m$ into G_i as follows:

If $x_i \longleftarrow x_j \vee x_k$ then G_i is $x_i \leftrightarrow (x_j \vee x_k)$

If $x_i \longleftarrow x_j \wedge x_k$ then G_i is $x_i \leftrightarrow (x_j \wedge x_k)$

If $x_i \longleftarrow \neg x_j$ then G_i is $x_i \leftrightarrow \neg x_j$

At this point, C is a satisfiable circuit iff $G_{n+1} \wedge \ldots \wedge G_m$ is a satisfiable formula. However, this formula is not in 3CNF; we convert $G_{n+1} \wedge \ldots \wedge G_m$ to 3CNF as follows:

If G_i is of the form $x_i \leftrightarrow (x_j \vee x_k)$, then convert it to G'_i given by:

$$(\neg x_i \vee x_j \vee x_k) \wedge (x_i \vee \neg x_j) \wedge (x_i \vee \neg x_k)$$

If G_i is of the form $x_i \leftrightarrow (x_j \wedge x_k)$, then convert it to G'_i given by:

$$(\neg x_i \vee x_j) \wedge (\neg x_i \vee x_k) \wedge (\neg x_j \vee \neg x_k \vee x_i)$$

If G_i is of the form $x_i \leftrightarrow \neg x_j$, then convert it to G'_i given by:

$$(\neg x_i \vee \neg x_j) \wedge (x_j \vee x_i)$$

Let $F_C = G'_{n+1} \wedge \ldots \wedge G'_m$.

3. Consider the language $L = \{0^i 1 0^j \mid 1 \leq i \leq j\}$. For example, the strings 010,0100,00100 are all in L, but the strings 10, 0010, 001100 are not.

Below is a partially specified Turing Machine M which is supposed to recognize L.

[4] (a) Suppose the input string is 00100, so the initial tape configuration is:

$$\texttt{...bb}\underline{\texttt{0}}\texttt{0100bb...}$$

and the head is reading the underlined symbol. Follow the computation of M on this input until M reaches state q_4 and the head is scanning the symbol 1. Give the resulting tape configuration below:

$$\texttt{...bb010bb...}$$

[4] (b) Complete the three indicated lines below by giving the action of M in each case:

	state	symbol	action
	q_0	0	q_1, print b, right
(input must start with 0)	q_0	b,1	q_{reject}
(go right to 1)	q_1	0	q_1, print 0, right
	q_1	1	q_2, print 1, right
	q_1	b	q_{reject}
(go right to b)	q_2	0	q_2, print 0, right
	q_2	b	q_3, print b, left
(two 1s)	q_2	1	q_{reject}
(cross out 0)	q_3	0	q_4, print b, left
(not enough zeroes to right)	q_3	1	q_{reject}
(go left to 1)	q_4	0	q_4, print 0, left
	q_4	1	q_5, print 1, left
	q_5	b	q_{accept}
	q_5	0	q_6, print 0, left
(go left to b)	q_6	0	q_6, print 0, left
	q_6	b	q_0, print b, right