Answer THREE questions, at least ONE from EACH of Sections A and B.

SECTION A

1. a. Give a complete definition of a standard, deterministic single-tape *Turing machine* (TM) and describe its operation. Define what it means to say that a TM *halts* for some input x.

[15 marks]

b. Define what it means to say that a TM *semidecides* a language L, and what it means to say that a TM *decides* a language L.

[5 marks]

c. Let $L \subseteq \{0,1\}^\star$ be the language

$$L = \{x \in \{0,1\}^\star : x \text{ is the binary representation of an odd natural number}\}.$$

Design a TM that semidecides L.

[6 marks]

d. Consider the statement: 'a language can be semidecided by a TM if and only if it can be decided by some other TM'. Is this statement true? Give a detailed explanation of your answer, including specific examples of languages where necessary.

[7 marks]

[Total = 33 marks]

2. Consider the following decision problem.

   Instance: Two Turing machines M_1 and M_2 and an input alphabet Σ_I.

   Question: Let $L_1 \subseteq \Sigma_I^\star$ and $L_2 \subseteq \Sigma_I^\star$ be the languages semidecided by M_1 and M_2 respectively. Is it the case that $L_1 \cap L_2 = \emptyset$ and $L_1 \cup L_2 = \Sigma_I^\star$?

   Prove from first principles that this problem is unsolvable. You may not assume the existence of other unsolvable problems and you may not rely on any results that you don't prove yourself.

   [Total = 33 marks]

3. a. Give a definition of a *tiling system*, and a definition of a *tiling*.

   [8 marks]

   b. Consider the problem P of deciding satisfiability of a closed predicate formula. Consider also the problem P' of deciding, given a tiling system T, whether or not a tiling exists. Give a complete definition of what it means in general to *reduce* some decision problem to another. Describe a reduction of P' to P and prove that it turns yes-instances of P' into yes-instances of P. You do not have to consider no-instances.

   [25 marks]

   [Total = 33 marks]

SECTION B

4. a. What is the *Hamiltonian Circuit Problem* (HCP)?

[6 marks]

Consider the following problem, called the ***figure-of-eight*** problem (FOE). An instance is an undirected graph $G = (V, E)$ with vertices V and edges E. G is a yes-instance if there is a sequence of vertices $(v_0, v_1, \ldots, v_{k-1})$ (some $k \geq 6$) such that

i. Each pair (v_i, v_{i+1}) is an edge (each $i < k - 1$) and (v_{k-1}, v_0) is an edge.

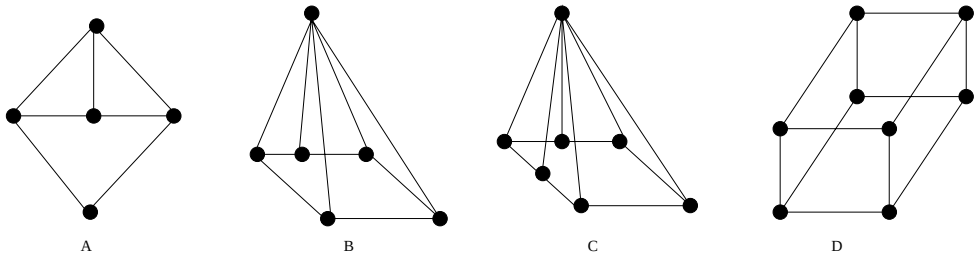
ii. Every vertex in V occurs at least once in the sequence.

iii. There is j with $2 < j < k - 2$ such that $v_0 = v_j$.

iv. No other vertex in the sequence is counted twice, i.e. if $v_s = v_t$ (any $s, t < k$) then either $s = t$ or $\{s, t\} = \{0, j\}$.

If there is no such sequence of vertices then G is a no-instance of FOE.

b. For each of the following graphs state whether it is a yes or a no-instance of FOE.



[12 marks]

c. Define a (p-time) reduction of HCP to FOE.

[15 marks]

[Total=33 marks]

5. a. Say what it means for one decision problem to reduce to another in p-time.

[6 marks]

b. Prove that p-time reduction is transitive.

[6 marks]

c. Let A_1 be a decision problem such that all its instances are no-instances and let A_2 be a decision problem where all its instances are yes-instances. Design a simple Turing machine that solves A_1 and a second simple Turing machine that solves A_2.

Design a third Turing machine that always terminates and leaves the three letters "YES" in the first three cells of the tape and blanks everywhere else. To define your Turing machines you can either draw state-transition diagrams or write down the set of states, alphabet, transition table etc., formally.

[10 marks]

d. Let A and B be decision problems and suppose $A \in \mathbf{P}$. Suppose further that B has at least one yes-instance and at least one no-instance. Prove that $A \leq_p B$.

[11 marks]

[Total=33 marks]

6. a. Describe how a *non-deterministic Turing machine* (NDTM) $M = (Q, \Sigma, q_0, \delta, F)$ works. Explain what it means when we say that a NDTM M solves the decision problem P. Make sure you include a clear definition of when M accepts an input and when it rejects its input.

[8 marks]

b. What is the run-time of a non-deterministic Turing machine T running on an input w?

[5 marks]

c. Define the class of non-deterministic, polynomial time problems ($\mathbf{NP}$).

[4 marks]

d. Let M be a non-deterministic Turing machine that runs in p-time. Write down a *deterministic* algorithm that emulates M. Your deterministic algorithm can be written in pseudo-code if you like.

[8 marks]

e. If the run-time of M is $O(n)$, give an estimate of the worst-case complexity of the deterministic emulator you designed in question 6.d.

[8 marks]

[Total=33 marks]

END OF PAPER