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

SECTION A

1. a. Give a general explanation of the way in which a decision problem can be represented as a formal language using an *encoding scheme*. Give three properties that any encoding scheme should have.

[8 marks]

b. An *integer matrix* is simply a matrix of any size having integer elements. Devise an encoding scheme that would be suitable for representing *arbitrary* decision problems concerning integer matrices as formal languages.

[5 marks]

c. Assume that there exists an encoding scheme for Turing machines, such that code(M) denotes a string over the alphabet $\{0, 1\}$ that represents a Turing machine M. Let L be the language defined as follows.

$$L = \{x : x = \text{code}(M) \text{ for some Turing machine } M \text{ and } M \text{ accepts } x\}$$

Prove, giving as much detail as you can, that L is not recursive.

[20 marks]

[Total 33 marks]

2. a. Let A be a formal language over an alphabet Σ, and let M be a Turing machine.

i. Define what it means to say that M *recognizes* A.

[3 marks]

ii. Define what it means to say that A is *recursive*.

[2 marks]

b. Let A and B be two formal languages. Define, giving as much detail as you can, what it means to say that A reduces to B.

[6 marks]

c. Let $A \leq B$ denote that A reduces to B.

i. Prove that if B is recursive and $A \leq B$ then A is recursive. Explain in detail what this allows you to conclude in the case that $A \leq B$ and A is not recursive.

[12 marks]

ii. What can be concluded if $A \leq B$ and A is recursive. Explain your answer.

[10 marks]

[Total 33 marks]

3. a. Define the *Empty Tape Halting Problem* (ETHP), and explain what it means to say that this problem is *unsolvable*.

[8 marks]

b. Prove, by reducing the empty tape halting problem or otherwise, that the following decision problem is unsolvable.

Instance: A Turing machine M, an input alphabet T, and a pair of symbols σ_1 and σ_2.

Question: Let $L(M)$ be the language over T accepted by M. Does $L(M)$ contain any string having $\sigma_1\sigma_2\sigma_1$ as a substring.

[15 marks]

c. Define the *Tiling Problem* and outline the way in which it can be proved to be unsolvable by reducing ETHP. You do not have to provide full technical details of the reduction, but should give a general explanation of the way in which it works.

[10 marks]

[Total 33 marks]

SECTION B

4. a. Define $n - SAT$.

[7 marks]

b. Show that $4SAT \leq 3SAT$, i.e. $4SAT$ reduces in p-time to $3SAT$.

[17 marks]

c. Hence show that $4SAT$ and $3SAT$ are p-time equivalent.

[9 marks]

[Total 33 marks]

5. a. Explain what a *non-deterministic Turing machine* (NDTM) is. In particular, you must be clear about when a NDTM accepts its input, when it rejects its input and when it is undefined on its input.

[11 marks]

b. A *composite number* n is a positive integer with a factor f such that $1 < f < n$. Define a NDTM that tests for composite numbers, written in a binary notation, in a reasonably efficient way. You may define your Turing machine by drawing a state-transition diagram for it or using a suitably precise pseudo-code, if you like. You may use, without further explanation, a deterministic procedure '$factor(m, n)$' to test if one number m is a factor of another n, and you may assume that the procedure '$factor$' runs in time $O(|m| \times |n|)$, where $|m|$ is the length of the binary representation for m. You may also use a procedure '$greater$' which runs in linear time and tests if one number is greater than another.

What, in order terms, is the run-time of your machine? What does this tell you about the complexity of the composite number problem?

[11 marks]

c. Suppose we use an alphabet with only one symbol, '1', apart from the blank symbol, so numbers must be represented in the unary notation. Define a *deterministic* Turing machine that runs in p-time and decides whether a number is prime or not. Again you may want to define your Turing machine by drawing a state-transition diagram or using pseudo-code for it and you may use the procedures '$factor$' and '$greater$' defined above. What, in order terms, is the worst-case run-time of your Turing machine?

[11 marks]

[Total 33 marks]

6. a. What do we mean when we say that a decision problem A *reduces in polynomial time* to another decision problem B (written $A \leq B$)?

[5 marks]

An instance of the *graph node colouring problem* (GNCP) is an undirected graph G and a set of colours C. (G, C) is a yes-instance of GNCP if and only if it is possible to colour each node of G with a single colour from C in such a way that no two adjacent nodes of G are given the same colour.

An instance of the *graph edge colouring problem* (GECP) is an undirected graph G and a set of colours C. (G, C) is a yes-instance of GNCP if and only if it is possible to colour each edge of G with a single colour from C in such a way that no two coincident edges of G are given the same colour.

b. Show that $GECP \leq GNCP$.

[18 marks]

c. Assume that GECP is NP-hard. What, if anything, can you deduce about the complexity of the graph node colouring problem, using your reduction from the previous part of this question? Justify your answer. You may assume that p-time reduction is transitive.

[10 marks]

[Total 33 marks]

END OF PAPER