NOTE TO STUDENTS: This file contains sample solutions to the term test together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly. We will make *no* exception to the marking scheme, unless you can clearly demonstrate that it is somehow incorrect.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

Question 1. [10 MARKS]

A *multi-headed Turing machine* (MHTM) is like a regular Turing machine except that it has k independent heads (working on the same, single tape), for some fixed $k > 1$. Initially, all heads are on the leftmost symbol of the input. For each transition, the current state and the symbols read by each head determine the next state, the symbols written by each head, and the direction of movement for each head—if multiple heads attempt to write different symbols on the same tape square, we adopt the convention that the symbol written is the one dictated by the latest numbered head (for example, if heads 2, 3 and 5 attempt to write symbols to a particular cell, the symbol written is the symbol dictated by head 5).

Prove or disprove that multi-headed Turing machines are equivalent to regular Turing machines.

SAMPLE SOLUTION:

We prove the equivalence of MHTMs and TMs, for any fixed $k > 1$.

- Given a TM M, construct an equivalent MHTM M' as follows: synchronize all heads of M' so that for each transition, they write the same symbol and move in the same direction as the one head of M. By construction, M' has the same behaviour as M on every input string.
- Given a MHTM M, construct an equivalent TM M' as follows: M' uses extra tape symbols to keep track of the position of M's heads on the tape (including the cases when more than one head are on the same square); every transition of M is simulated by two passes of M' over the tape: one to read and remember the symbols under M's heads, and one to update the tape according to M's transition (in a way similar to the simulation of a multi-tape TM done in class). By construction, M' has the same behaviour as M on every input string.

MARKING SCHEME:

A. 2 marks: attempt to prove equivalence
B. 4 marks: argument that MHTMs can simulate TMs
C. 4 marks: argument that TMs can simulate MHTMs

Question 2. [10 MARKS]

Show that the language

$$L = \{\langle M\rangle \in \{0,1\}^* : M \text{ is a TM such that } L(M) \text{ contains at least one string that contains a } 0\}$$

is recognizable. (It is sufficient to give a detailed high-level answer.)

SAMPLE SOLUTION:

The following is a recognizer for L.
On input $\langle M\rangle$, use dovetailing to simulate M on all input strings that contain a 0 for all numbers of steps, until M accepts at least one string. More precisely:

1. Extract the input alphabet Σ_M from $\langle M\rangle$ and check that $0 \in \Sigma_M$, reject if it isn't.
2. Enumerate all strings over the input alphabet of M that contain a 0: call these strings $s_1, s_2, s_3, \ldots$.
3. For $i = 1, 2, 3, \ldots$:
4. Simulate M for at most i steps on each of the input strings $s_1, s_2, \ldots, s_i$, one at a time.
5. Accept immediately if M enters the accept state during any of these simulations.

If $\langle M\rangle \in L$, then this recognizer accepts $\langle M\rangle$ because it will eventually simulate M on enough inputs and for enough steps to find one input that is accepted by M.
Conversely, if $\langle M\rangle \notin L$, then this recognizer will never accept $\langle M\rangle$ because it will continue forever simulating M on more and more strings for more and more steps.

MARKING SCHEME:

A. 2 marks: attempt to describe a recognizer for L (not required to be a TM—any clear algorithm is acceptable)
B. 2 marks: recognizer uses dovetailing
C. 3 marks: recognizer accepts all $\langle M\rangle \in L$
D. 3 marks: recognizer does not accept any $\langle M\rangle \notin L$

EXPECTED ERRORS:

E. -1 mark per occurrence (to a maximum of -4): confusing/incorrect use of notation (*e.g.*, recognizer works on input $\langle M, w\rangle$)

Question 3. [10 MARKS]

Show that the language

$$L = \{\langle M\rangle \in \{0,1\}^* : M \text{ is a TM such that } L(M) \text{ contains at least one string that contains a } 0\}$$

is undecidable. (Your answer will be marked on its structure as well as its content.)

SAMPLE SOLUTION:

For a contradiction, assume that R is a decider for L. Construct a decider S for A_{TM} as follows.
S = "On input $\langle M, w\rangle$:

1. Compute the description of the following TM:
   M' = 'On input x: Run M on w (ignore x) and do the same.'
2. Run R on $\langle M'\rangle$ and do the same."

Then, S is a decider (because R is a decider) and S accepts $\langle M, w\rangle$ iff R accepts $\langle M'\rangle$ iff M' accepts at least one string that contains a 0 iff M' accepts every string (by construction, M' either accepts every string or no string) iff M accepts w.
Hence, S decides A_{TM}, a contradiction.

MARKING SCHEME:

- 5 marks: form
  - A. 1 mark: "assume R decides L"
  - B. 1 mark: "construct decider S for A" (where A is known to be undecidable)
  - C. 2 marks: clear description of S (correct form of input; correct use of notation throughout)
  - D. 1 mark: "S accepts iff input belongs to A"
- 5 marks: content
  - E. 3 marks: correct decider S (including correct use of R)
  - F. 2 marks: correct argument that S is a decider that accepts iff input belongs to A

ALTERNATE SAMPLE SOLUTION:

We show that $A_{TM} \leq_m L$. Since A_{TM} is undecidable, this proves L is also undecidable.
Given $\langle M, w\rangle$, construct $\langle M'\rangle$ as follows:
M' = "On input x: Run M on w (ignore x) and do the same."
Clearly, this transformation is computable. Also, if M accepts w, then M' accepts every string (so it accepts at least one string that contains a 0), and if M does not accept w, then M' accepts no string (so it does not accept at least one string that contains a 0). Hence, $\langle M, w\rangle \in A_{TM}$ iff $\langle M'\rangle \in L$.

Alternate Marking Scheme:

- 5 marks: form
  - A. 1 mark: "show $A \leq_m L$" (where A is known to be undecidable)
  - B. 2 marks: clear description of mapping (correct form of input; correct use of notation throughout)
  - C. 1 mark: "this is computable"
  - D. 1 mark: "$x \in A$ iff $f(x) \in L$"
- 5 marks: content
  - E. 3 marks: correct mapping
  - F. 2 marks: correct argument that $x \in A$ iff $f(x) \in L$