CITS2211 Discrete Structures
Week 12 Exercises – Turing Machines

For an implementation-level description of a Turing Machine you do not have to give a formal specification of the machine’s moves, but only to explain the algorithm in English prose in terms of the series of steps the machine would use for this calculation.

1. Give a brief explain for the following terms in the context of Turing Machines: state, input tape, move, halt, halting problem, recognise (a language), compute (a function).

Solution:

(a) State: As in a finite state machine, a value from a finite set of state identifiers that defines part of the current context of the TM.

(b) Input tape: An infinite tape on which is written a sequence of blanks and symbols a from the given alphabet for the TM.

(c) Move (of the TM): Given a transition specified by (s,i,o,s’,D) a TM in state s looking at input symbol i will write to the tape, change to state s’ and move in direction D (either one space left or right).

(d) Halt: The TM reaches a state and input tape symbol for which there is no move defined

(e) Halting Problem: Is there an algorithm that given a string $s_T$ representing a Turing machine $T$, and an input string $\alpha$ can determine whether $T$ will halt if it is given the string $\alpha$ as input?

(f) Recognise: A TM recognises language $L$ if given a word $w \in L$ on its input tape, the TM will halt in an accepting state.

(g) Compute: A TM computes a function $f(x) = y$ if given an input tape with a representation of $x$ it produces an output tape with a representation of $y$ (and usually then halts).
2. Outline an *implementation-level description* of a Turing machine that recognises the language $(10)^*$. State any assumptions you make.

**Solution:** Start at the leftmost 0. Assume we can stop at the right hand end. Or you can choose to move back to the left hand end of the string after reaching the rightmost blank (i.e. the start position). Assume that if the machine encounters an input symbol with no move then it halts and rejects the string; we do not explicitly show transitions to a reject state.

(a) If blank then accept else continue
(b) Check for 1 and move R
(c) Check for 0 and move R
(d) Repeat from step 1

3. Write a state machine version of your Turing machine to recognise the language $(10)^*$ (as in the previous question). Show all the moves of this machine.

**Solution:**

![State Machine Diagram]

4. Outline an *implementation-level description* of a Turing machine that computes the function $f(n) = n - 2$ where a natural number $n$ is represented in unary.

**Solution:** Assume the number is represented in unary with 1 representing 0, 11 for 1, 111 for 2 etc.

(a) Machine starts at the left of the tape over the start symbol to the left of the first 1.
5. Outline an implementation-level description of a Turing machine that acts as a “doubler”. For Input: A string of 1s of length $n$ and for Output: A string of 1s of length $2n$. This machine calculates the function $f(n) = n + n$ where a natural number $n$ is represented in unary (with 0 being represented as an empty sequence of 1s).

(If on the other hand we regard the input as being in unary, but with 0 represented using a single 1, then this TM can be regarded as calculating the function $f(n) = 2(n - 1) + 1$. For example, three 1s will represent 2; doubled, we get six 1s, which represents 5; and $2(3 - 1) + 1 = 5$.)

**Solution:**
Here is one strategy: we mark our “input” string as “processed” by changing 1s to 0s, and we write out “output” just after the input (separated from it by a blank). Once our input is all gone, we move to the start of the output, and we are done.

(a) The R/W head should be over the start of the left-most input.
- If we see a blank, then we are all done: move right until we see a blank (the separator), then move right again, and we’ll be over the start of the output (which is possibly empty), so we accept.
- If we see a 1, then there is still work to do. Change the 1 to a 0.

(b) Go right to the right-hand edge of the input; go right over the blank; and if there are any 1s following, go right over those too.

(c) Move right one cell, and write “11” at our current position (i.e. just past the edge of the output) – that is, write a 1, then go right and write another 1.
(d) Go left until we see a 0, then move right and return to step 1.

There are many more strategies – if you have another, feel free to suggest it in the help forum.

6. Outline an implementation-level description of a Turing machine that computes the function \( f(x, y) = \max(x, y) \) where \( x \) and \( y \) are represented in unary number notation separated by a separator symbol *.
The machine starts at the leftmost non-blank cell. It should leave the tape containing \( z \) in unary, where \( z \) is the maximum of \( x \) and \( y \). Do not use any symbols other than 1, the separator * and blank at any time.

**Solution:**

Assume that one of the numbers is strictly larger than the other i.e. they are not equal. The algorithm used for the machine relies of the fact that \( \max(x, y) = 1 + \max(x - 1, y - 1) \). Move left and right crossing off matching 1s from the ends of the tape. Eventually there is a 1 on the left or the right, but no more 1s on the other side.

This problem is easier if you allow extra symbols. But since we only have * and 1 we must first mark the edges of the input tape so that we are able to restore the 1s for the maximum number side once we know it.

The strategy is:

**Part 1: mark the edges of the inputs on the tape**

(a) Machine starts at the left of the tape; Change the first 1 to a * to mark the left hand end.

(b) Move to the far right hand end and change the last 1 to a * to mark the right hand end.

(c) Move back left over the middle * until the left hand *.

**Part 2: cross off matching 1s either side of the middle until you run out of 1s on one side**

(a) Move right until the first 1, change it to a b and move left back to the middle *.
(b) If you find only blanks before the ∗ (no more 1s) then the right hand side is smaller and max is the left hand side. Change state to L, move left back to the middle and transition to part 3.

(c) Move left until the first 1, change it to a b and move left back to the middle ∗.

(d) If you find only blanks before the ∗ (no more 1s) then the left hand side is smaller and max is the righthand side. Change state to R, move right back to the middle and transition to part 3.

**Part 3: restore all the 1s on the maximum number side**

(a) If state is L then move left replacing every b or 1 with 1 until you reach the leftmost ∗. Replace that with 1 and halt.

(b) If state is R then move right replacing every b or 1 with 1 until you reach the rightmost ∗. Replace that with 1 and halt.