CS3052 Practical 1 Report

20007413

1 Task 2 - Turing Machines

1.1 TM 1 - Divides

This first Turing Machine (`divides.tm`) is designed to test whether or not a number divides another number (aka. whether or not the number is a factor).

The syntax for the tape is similar to the mathematical notation for divides: $a|b$, but here numbers are written as x number of `1`s (see. `configs/divides.tape` for an example).

My approach to this problem was to continuously subtract a from b one at a time, until there is no more b, meaning either (1) a full a has just been subtracted, meaning a can be multiplied to get b, or (2) a partial a has been subtracted from b, meaning a does not multiply to get b. This tells us whether a is a factor of b, which is also the requirement for whether a divides b.

1.2 TM2 - Prime

After completing the divides Turing Machine, I realized that this could be extended to checking whether a number is prime, as the definition of a prime number is having only 1 and itself as factors, meaning verification could be done by repeatedly checking whether the numbers 2-N divide N, and if none do, then N is prime.

For this, I extended the previous Turing Machine in a new file `isprime.tm`.

The format for the tape here requires some setup beyond just writing the number to be checked. Instead, the format is $\#11|N$, where N is written in the same way as before, written as N number of `1`s. The reason for this format is because the machine checks whether each number X, starting from 2 (written 11) divide N.

There are two ways for this to halt:

1. One of the divisible checks returns true, and $X \neq N$. This means that X is a factor N, and thus N cannot be prime

2. One of the divisible checks returns true, and $X = N$. This means that all of the numbers $< N$ are not factors of

N, thus meaning N is prime.

The format for this is writing $\#11|n$ where **n** is the number checked written as a series of 1s as before (see `configs/prime.tape` for an example).