# Exercise 2

Describe a TM that accepts its input string if the length of the string is prime. You do not have to mention all the transitions, but describe the design in enough detail so that if someone paid you enough, you could.

The TM will be based on the sieve of Eratosthenes algorithm.

The description of the algorithm is the following:

* On input x:
  1. Check that the first two tape cells after the left end-marker do not contain U. If they contain U then reject.
  2. Arrive at the end of the string and replace last symbol with this symbol $.
  3. Go all the way left on the tape
  4. Delete the first symbol
  5. Go to the right of the tape accenting “ ’ ”all empty symbols
  6. Mark the first non-empty tape and mark it with an “^”
  7. Shift all accented symbols to the right of the marked tape
  8. Shift the marked symbol to the right of the now accented symbols
  9. Delete the symbol and mark it with “^”
  10. Repeat step 7 to 9 until:
      + If the symbol $ is marked then reject
      + If the symbol $ is passed then go all the way left and repeat from step 5 to 10
        - If symbol $ is the first empty symbol when performing step 6 then accept

# Exercise 3

Describe a TM that accepts the set { a n | n is a power of 2 }, viewed as a language over the alphabet {a}. You do not have to mention all the transitions, but describe the design in enough detail so that if someone paid you enough, you could.

The TM will behave in a similar way as the algorithm above.

The description of the algorithm is the following:

* On input x:
  1. Check that the input is not an empty string. In that case reject
  2. Arrive at the end of the string and replace the last symbol with this symbol $.
  3. Go all the way left on the tape
  4. Accent the first symbol
     + If the first symbol is $ then accept
  5. Mark the second symbol with an “^”
  6. Shift the accent symbols to the right of the marked symbol
  7. Shift the marked symbol to the right of the accent symbols
     + If $ is that symbol then accept
     + If an empty symbol is marked then reject
  8. Go left and accent all symbols
  9. Repeat step 6,7,8

# Exercise 4

## Prove that recursively enumerable sets are closed under union and intersection

**Closed under union:**

Let A be a recursive enumerable set. Let B be a recursive enumerable set.

Thus, there exists a Turing machine M such that L(M)=A and a Turing machine M’ such that L(M’)=B.

We can create a Turing machine N which operates on the following way:

* On input x:
  + Run alternatively machine M and machine M’
  + If either machine M or machine M’ accept then accept.
  + If both machine M and machine M’ reject then reject.

It is trivial to see that the language accepted by the above Turing machine is the following:

L(N) = L(M) ∪ L(M’)

It is also trivial to see that if one machine (either M or M’’) loops on input x and the other machine rejects x or loops on x as well then machine N will loop on x as well.

Therefore, the set L(N) is recursive enumerable because N will accept all strings in L(N) but may loop on other strings that do not belong to L(N).

**Closed under intersection:**

Let A be a recursive enumerable set. Let B be a recursive enumerable set.

Thus, there exists a Turing machine M such that L(M)=A and a Turing machine M’ such that L(M’)=B.

We can create a Turing machine N which operates on the following way:

* On input x:
  + Run alternatively machine M and machine M’
  + If both machine M and machine M’ accept then accept.
  + If either machine M or machine M’ reject then reject.

It is trivial to see that the language accepted by the above Turing machine is the following:

L(N) = L(M) ∩ L(M’)

It is also trivial to see that if both machine loop on x then N will loop on x as well. It is also trivial to see that if either machine M or machine M’ accept and the other machine loops then N will loop as well.

Therefore, the set L(N) is recursive enumerable because N will accept all strings in L(N) but may loop on other strings that do not belong to L(N).

## Prove that recursive sets are closed under union and intersection

**Closed under union:**

Let A be a recursive set. Let B be a recursive set.

Thus, there exists a Total Turing machine M such that L(M)=A and a Total Turing machine M’ such that L(M’)=B.

We can create a Turing machine N which operates on the following way:

* On input x:
  + Run alternatively machine M and machine M’
  + If either machine M or machine M’ accept then accept.
  + If both machine M and machine M’ reject then reject.

It is trivial to see that the language accepted by the above Turing machine is the following:

L(N) = L(M) ∪ L(M’)

Besides, L(N) is a recursive set because on any input x both M and M’ will either accept or reject.

**Closed under intersection:**

Let A be a recursive set. Let B be a recursive set.

Thus, there exists a Total Turing machine M such that L(M)=A and a Total Turing machine M’ such that L(M’)=B.

We can create a Turing machine N which operates on the following way:

* On input x:
  + Run alternatively machine M and machine M’
  + If both machine M and machine M’ accept then accept.
  + If either machine M or machine M’ reject then reject.

It is trivial to see that the language accepted by the above Turing machine is the following:

L(N) = L(M) ∩ L(M’)

Besides, L(N) is a recursive set because on any input x both M and M’ will either accept or reject.

There is another way to prove that recursive sets are closed under union and intersection.

Since A and B are recursive then A and B are r.e. and Ac and Bc are r.e as well.

We know that if A ∪ B is recursive then A ∪ B must be r.e and (A ∪ B)c must be r.e as well.

A ∪ B is r.e as it follows from the fact that r.e sets are closed under union.

(A ∪ B)c = (Ac ∩ Bc) which by the closure properties of r.e sets is r.e as well.

Consequently, recursive sets are closed under union.

A similar proof can be derived for proving that recursive sets are closed under intersection.

## Further notes:

It is necessary in the first machine N built for showing that recursive enumerable sets are closed under Union that M and M’ are run alternatively. Indeed, it is enough that just one of them accept to accept the string x. However, if we run first a machine and subsequently the other, it may happen that the first machine loops on x and so we cannot find out whether the other machine accepts.

For all the other machines built above, it is not necessary that M and M’ are run alternatively.