4.1

a. The relation defined is partial order on the set $\sum^*$ if it is transitive, antisymmetric and reflexive. Hence need to show 3 different cases:

i. Reflexive: To show it is reflexive p , we need to show p is a prefix of p, this is possible when we consider p*w=p and where w is essentially an empty set. The relation is reflexive if for all elements because p will always be a prefix of p, Eg. Let the word be “University ”We can essentially take the prefix as the whole word p= University, in this case this will always be reflexiv since we are taking w as an empty set.

ii. Antisymmetric: To show that the relation is antisymmetric, :
If p is a prefix of q, $p \leqslant q$ E w1
and $q \leqslant p$ E w2
then pw1=q
qw2=p
then by antisymmetry p=q

iii.Transitivity: For transitivity we have to show if $p \leqslant q$ and $q \leqslant r$ then $p \leqslant r$.

It doesn’t matter how you break up p and q, pq will always be the same. w1= pq, pq=w2 , therefore w1=w2.

b. A relation is a strict partial order on the set $\sum^*$ if it is irreflexive, asymmetric and transitive.

1. Irreflexive: To show that it is irreflexive we can do it by providing an example. Since p≠w , the prefix p will always be different than w. Consider the example ‘COMP’ , CO $<$ COMP (co is a prefix of comp) , however COMP cannot and will never be a prefix of CO. It is logically flawed since both of them cannot be equal to each other to satisfy (a,a) => reflexivity, hence irreflexive.

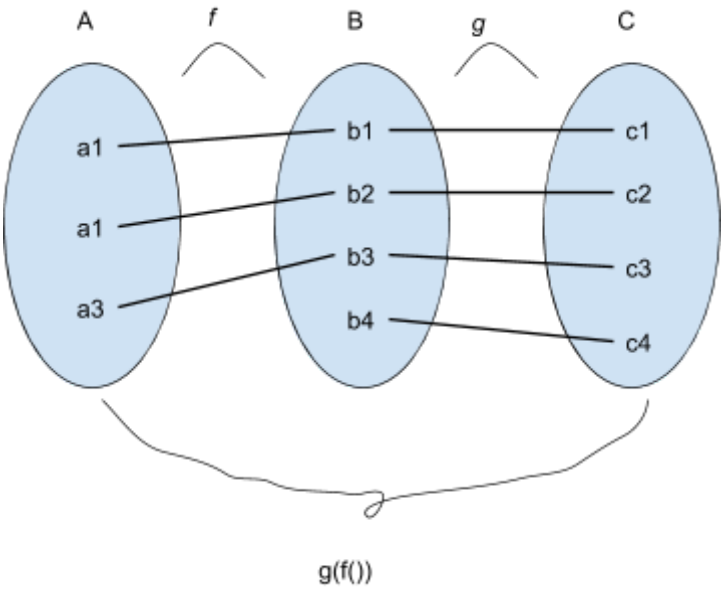
2. Suppose $p < p$, where pw= p
then $p < q$, assume this is true
now $q < p$, q cannot be the prefix of p , this is only possible if they are equal. And it cannot be equal , because of the condition in the question p≠w , hence asymmetric.

3. To show it is transitive it is quite similar to how we showed it in part a,
   To show,p $\prec$ q $\prec$ r therefore p $\prec$ r
   In general if we have,
   pk1=q
   and qk2=r
   Hence pk3=r
   Take the example,
   ab $\prec$ abc , abc $\prec$ abcd , ab $\prec$ abcd , hence shown.

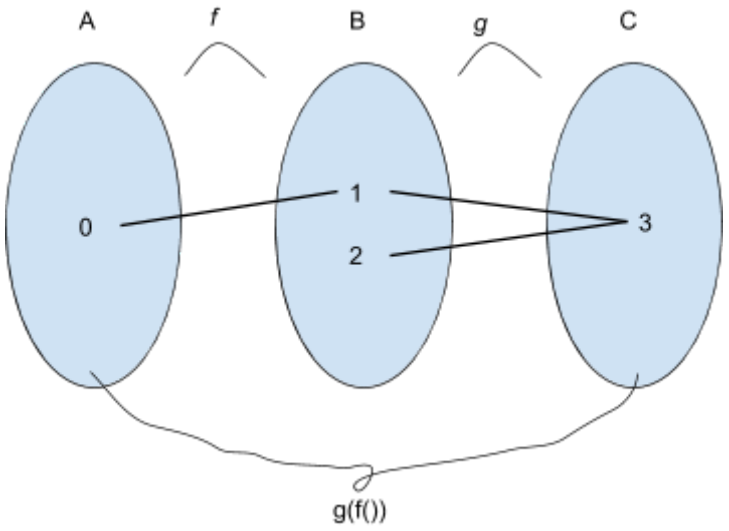
C. The two relations are also total because, to show that it is total we only have to show that one of them is true because the total is defined as $\forall a, b \in A.(a, b) \in R \vee (b, a) \in R$, so A or B.If one of them is true, it is total. So either p $\preccurlyeq$ w **or** p $\prec$ w is true. One of the two has to be true if p=w then p $\preccurlyeq$ w is true etc.

4.2

a. Let A, B and C be sets and let f : A → B and g : B → C be two functions. If g(f()) is bijective we have to prove *f* is injective and *g* is surjective. This can be best shown through graph notation.



The composite function is bijective because every element of A or the domain of g(f()) maps to only one element of co-domain C. Now *f* is injective because every element of co domain B is mapped by **at most** one element from domain A. *g* is surjective because every element of co domain C is mapped by **at least** one element of domain B. Finally if we zoom out and look at g(f()) for domain A and co domain C is mapped by **exactly** one element . Hence shown.



b. Consider the graph notation example *f* is injective because at most one element of domain A is mapped to every element of co-domain B . *g* is surjective because at least one element is mapped onto co domain C. Now g(f()) is not bijective because 3 is mapped by more than one element.

c. Let's take an example of a function *f(x) = x^2 , f(x)* is not surjective because it is mapped by 2 elements, a positive and a negative number gives same output. *g(x)= sqrt x* is not injective because it is undefined for negative values. g(f(x)) is x which is bijective since it is a straight line and a one to one function.

3. isSpecialPrime :: Integer -> Bool
isSpecialPrime num
 |elem num list == True = True
 |otherwise = False
 where
  list= filter isPrime (map (+1) (zipWith(+) (1:filter isPrime [1..num]) (filter isPrime[1..num])))

isPrime :: Integer -> Bool
isPrime x
 |x ==1 = False
 |length([a|a <- [2..(div x 2)], (mod x a) ==0])== True

My function works perfectly, the only problem is that it prints 1 as well in the list. Could you please tell me why it's doing that? And how you would fix it.
Otherwise, the logic is I created two lists of primes , the other list has an extra element so that we can align it up. What the zipWith does is that it adds the 2 lists of prime together, so now the map function adds 1 to each element of the new list cuz it is stated in the question.

Finally the last filter isPrime, filters out the primes from the non primes. Elem checks if the number is in the list or not , otherwise returns false.