```haskell
-- Model solution for Tut11
--
-- Copyright [2000..2001] Gabriele Keller

substring:: String -> String -> Bool
substring ""    str  = True
substring str   ""   = False
substring str1  str2 =
  (prefix str1 str2) || (substring str1 (tail str2))

prefix:: String -> String -> Bool
prefix "" str = True
prefix str "" = False
prefix (s1:str1) (s2:str2)
  | s1 == s2  = prefix str1 str2
  | otherwise = False


{-

  Question 1
  ----------

  First, we derive the timing function for T_p (T_prefix)

  If one of the input strings is empty, only one basic step
  is necessary:

  T_p (0,m) = 1
  T_p (n,0) = 1

  Otherwise, depending on the result of the comparison, either
  T_p (n,m) = 2 + T (n-1,m-1)    (1)
  T_p (n,m) = 2                  (2)

  Since we have to consider the worst case, we use (1) =>

    T_p (n,m) = 2 * min (n,m) + 1

  T_s (substring)

  T_s (0,m) = 1
  T_s (n,0) = 1

  (3 func. applications, 1 boolean operation), worst case
  T_s (n,m) = 4 + T_p (n,m) + T_s (n, m -1)
            = 4 + 2 * min (n,m) + 1 + T_s (n, m -1)
            = 5 + 2 * min (n,m) + T_s (n, m -1)
  =>
  To find the exact solution, we have to distinguish between two cases:

  if m <= n:
  T_s (n,m) = sum (i=1)(m) (5 + 2 *  i)
    =>
  T_s (n,m) = 5m + m^2 + m = m^2 + 6m  +1

  m > n  (for m-n steps, m >= n in the rec. call)
  T_s (n,m) = sum(i=n+1)(m) (5 + 2 * n) + T_s(n,n)
            = (5 + 2n)(m-n) + n^2 + 6n +1
            = 2mn + 5m - n^2 + n +1

  According to the observations discussed in the lecture, T_s(m,n) is
  in the same O-class as T(m,n) = mn if m > n

  Question 2
  -----------

  To prove that a function T is *not* in O(f), we have to show that
  no constant value c exists such that T is less than c * f for *all*
```

```
values above a certain point.
Proof by contradiction: we assume c exists, we show that by incrementing
the arguments sufficiently, T 'overtakes' f again

Question 2 - 1

T_s (m,n) is not in O(m), since there are no constants c n0 and m0
such that

   m * n < c * m   for all m > m0, n > n0

(assume there were, then

   m * n < c * m for all n > n0
=>     n < c     for all n > n0, which is not
                 true ifor n > c)


Question 2 - 2

T_s (m,n) is not in O(m), since there are no constants c n0 and m0
such that

   m * n < c * n * n  for all m > m0, n > n0

(assume there were, then

   m * n < c * n * n for all n > n0
=>     m < c * n     for all n > n0, m > m0
                     which is not
                     true if we choose m big enough,
                     i.e, m > (n0+1) * c



Question 3

T is in O(m*n)

-}
```