```
(* $Id: ackermann.ml,v 330.3 2003-02-03 10:42:46-08 - - $ *)
(* Ackermann's Function *)

(* tuple version *)
let rec ackt = function
   | ( 0, 0, k ) -> k
   | ( 0, j, k ) -> ackt( 0, j - 1, k ) + 1
   | ( 1, 0, k ) -> 0
   | ( i, 0, k ) -> 1
   | ( i, j, k ) -> ackt( i - 1, ackt( i, j - 1, k), k )
   ;;

(* curried version *)
let rec ackc i j k = match (i, j) with
   | ( 0, 0 ) -> k
   | ( 0, j ) -> (ackc 0 (j - 1) k) + 1
   | ( 1, 0 ) -> 0
   | ( i, 0 ) -> 1
   | ( i, j ) -> ackc (i - 1) (ackc i (j - 1) k) k
   ;;

(*
 * Prove that:
 *         ack 0 j k = k + j
 *         ack 1 j k = k * j
 *         ack 2 j k = k ** j
 * What is ack 3 j k ?
 *)
let add = ackc 0;;
let mul = ackc 1;;
let exp = ackc 2;;
let ttt = ackc 3;;

let inc = add 1;;
let dbl = mul 2;;
let sqr x = mul x x;;

(*
* More usual version of Ackermann's function,
* using only two parameters.
*)
let rec ak m n = match (m, n) with
    | (0, n) -> n + 1
    | (m, 0) -> ak (m - 1) 1
    | (m, n) -> ak (m - 1) (ak m (n - 1))
    ;;
```