```
(*
 Si consideri la Sintassi Astratta sotto, per il Lambda Calcolo:
        Lambda::= Var | Const | [$] Var Lambda | [@] Lambda Lambda
 dove usiamo $ per l'astrazione e @ per l'applicazione.
 Si chiede di fornire in Ocaml strutture, tipi, operazioni per la
 costruzione, l'accesso e la presentazione degli Abstract Syntax Tree
 ...
 Si mostri poi, come e' costruito il termine:
            [$ - ([x], [$ - ([y] [@ - ([x], [y])])]),
 e utilizzando printAST, se ne mostri la presentazione fornita dal
 sistema definito.
*)

(*
 Questo modulo e' il corrispettivo del modulo ain della versione
 fornita in C, vedi modulo main
*)

open Printf;;                (* un modulo di OCaml con operazioni per ... *)

#use "LastH.ml";;
#use "LastE.ml";;

let x = mkVar "x";;
let y = mkVar "y";;
let xy = mkApp x y;;
let t = mkAbs "x" (mkAbs "y" xy);;
printf "Il termine richiesto e':\n     %s\n" (printAST t);;


(*
# #use "MainE.ml";;
type last =...
...
val t : last = Abs ("x", Abs ("y", App (Var "x", Var "y")))
Il termine richiesto e':
     [$-([x],[$-([y],[@-([x],[x])])])]
- : unit = ()
*)
```