fun identity (x) = x

(* Here's a pretty boring function *)
(* it's going to have type 'a → 'a *)
(* it takes anything, and returns that same type *)

fun full_typed_identity (x : α) : α = x

(* You've seen type variables in types *
* you can write them yourself *
* this is exactly the same as above *)

datatype Option = Some α

(* You can also use type variables in datatype bindings *
* polymorphic datatype bindings *)

fun map (f, xs) = case xs of [] ⇒ [] | x :: xs' ⇒ (f x) :: map (f, xs')

fun double_all xs = map (λ(x) ⇒ 2 * x, xs)

fun double x = (* anonymous functions *)
(* we've seen function bindings like the following *)

val doubled = double_all [1, 2, 3, 4, 5]

fun replicate x = (* anonymous functions *)
(* we've seen function bindings like the following *)

let val v1 = apply_to_five (λ(str) ⇒ str "Hola" ^ str)
val v2 = apply_to_five (λ(str) ⇒ str "Bonjour" ^ str)
val v3 = apply_to_five (λ(str) ⇒ str "Hello" ^ str)
val v4 = apply_to_five (λ(str) ⇒ str "Hallo" ^ str)

fun add_greeting Spanish = (λ(str) ⇒ str "Hola" ^ str)
fun add_greeting French = (λ(str) ⇒ str "Bonjour" ^ str)
fun add_greeting English = (λ(str) ⇒ str "Hello" ^ str)
fun add_greeting German = (λ(str) ⇒ str "Hallo" ^ str)

val es_greet = add_greeting Spanish
val fr_greet = add_greeting French
val en_greet = add_greeting English
val de_greet = add_greeting German

val str1 = es_greet "Mundo"
val str2 = fr_greet "Monde"
val str3 = en_greet "World"
val str4 = de_greet "Welt"

fun filter(p, xs) = case xs of [] ⇒ [] | x :: xs' ⇒ if p x then x :: filter(p, xs') else filter(p, xs')

fun count_zeros xs = fold((λ(acc, x) ⇒ if x=0 then acc+1 else acc), 0, xs)

type date = int × int × int

fun day (d : date) = #1 d
fun month (d : date) = #2 d
fun year (d : date) = #3 d

(* homework 1 revisited *)
(* 2 *)
(* count how many dates in a list are in the given month *)
fun is_in_month((_,m,_), month) = (m = month);
fun number_in_month(dates, month) = let
  fun check_date d = is_in_month(d, month)
  in
  fold (check_date, dates)
  end;

(* 3 *)
(* count how many dates in a list are in the given list of months *)
fun count_zeros xs = fold((λ(acc,xs) ⇒ if x=0 then acc+1 else acc), 0, xs)
fun count_zeros xs = fold((λ(acc,xs) ⇒ if x=0 then acc+1 else acc), 0, xs)
fun number_in_months(dates, months) =
let
  fun get_month_count m = number_in_month(dates, m)
in
  sum(map(get_month_count, months))
end;

(* 5 *)
(* return list of dates that are in given list of months *)
(* could be done with dates_in_month and flat_map *)

(* 6 *)

fun n_times (f,n,x) = 
  if n=0
    then x
  else f (n_times(f,n-1,x))

fun get_nth(xs, n) = hd (n_times(tl, n-1, xs));