SCHRECK MP2I
Corentin

DM d’informatique n°3

```
(*exo 30*)
let plusetfois x y = [x+y;x*y];;
plusetfois 1 2;;
(*[3;2]*)
plusetfois 0 4;;
(*[4;0]*)

(*exo 31*)
let ppp2 n =
    let rec recppp2 p =
      if p>=n then p else recppp2 (2*p)
    in recppp2 1;;

ppp2 0;;
(*1*)
ppp2 1;;
(*1*)
ppp2 16;;
(*16*)
ppp2 42;;
(*32*)

(*exo 32*)
let leeloo l =
  let rec getelement t n = match t with
    |[] -> failwith "wrong list size"
    |a::b -> if n=0 then a else let m=n-1 in getelement b m
  in getelement l 4;;

let l1=[];;
leeloo l1;;
(*failure wrong list size*)
let l2=[1;2;3];;
leeloo l2;;
(*failure wrong list size*)
let l3=[1;2;3;4;5;6;7;8];;
leeloo l3;;
(*5*)




(*exo 33*)
let coupe l=
  let rec split l1 l2 l3 = match l1 with
```

SCHRECK MP2I
Corentin

```
    |a::b::c -> split c (l2@[a]) (l3@[b])
    |a::b -> (l2@[a],l3)
    |[] -> (l2,l3)
  in split l [] [];;

coupe l1;;
(*([],[])*)
coupe l2;;
(*([1;3],[2])*)
coupe l3;;
(*([1;3;5;7],[2;4;6;8])*)

(*exo 34*)
let colle l1 l2 =
  let rec merge l3 l4= match (l3,l4) with
    |([],[]) -> []
    |(a::b,c::d) -> [a;c]@merge b d
    |_ -> failwith "wrong list size"
  in merge l1 l2;;

let l4=[1;3;5];;
let l5=[2;4;6];;
colle l4 l5;;
(*[1;2;3;4;5;6]*)
colle l1 l1;;
(*[]*)

(*exo 35*)
let compte l n =
  let rec nbelements l1 t = match l1 with
    |[] -> t
    |a::b -> if a=n then nbelements b t+1 else nbelements b t
  in nbelements l 0;;

let l6=[1;2;3;3;2;3];;
compte l6 1;;
(*1*)
compte l6 2;;
(*2*)
compte l6 3;;
(*3*)
compte l1 42;;
(*0*)


(*exo 36*)
let sommesi l predicat=
  let rec ifsum l1 n = match l1 with
    |[] -> n
```

```
    |a::b -> if predicat a then ifsum b n+a else ifsum b n
  in ifsum l 0;;

let predicat1 a = match a with
  |3 -> true
  |n -> false;;
sommesi l6 predicat1;;
(*9*)
let predicat2 a = match a with
  |n when (n<6) -> true
  |_ -> false;;
sommesi l3 predicat2;;
(*15*)

(*exo 37*)
let majorpred l predicat =
  let rec testpred l1 nbtrue nbfalse = match l1 with
    |[] -> if nbfalse>nbtrue then false else true
    |a::b -> if predicat a then testpred b (nbtrue+1) nbfalse else testpred b nbtrue (nbfalse+1)
  in testpred l 0 0;;

majorpred l2 predicat1;;
(*false*)
majorpred l6 predicat1;;
(*true*)
majorpred l3 predicat2;;
(*true*)
majorpred l1 predicat2;;
(*true*)
(*0 éléments repsectent le prédicat et 0 éléments ne le respectent pas, donc au moins la moitié de la
liste le respecte*)

(*exo 38*)
let lexico l1 l2 =
  let rec compare l3 l4 = match (l3,l4) with
    |(a::b,c::d) when (a<c) -> true
```

SCHRECK MP2I
Corentin

```
    |(a::b,c::d) when (a=c) -> compare b d
    |([],a::b) -> true
    |_ -> false
  in compare l1 l2;;

lexico l1 l2;;
(*true*)
lexico l2 l1;;
(*false*)
lexico l6 l5;;
(*true*)
lexico l5 l6;;
(*false*)

(*exo 39*)
let aplatir l =
  let rec cat l1 = match l1 with
    |[] -> []
    |a::b -> a@cat b
  in cat l;;
let l7=[l1;l2;l3;l4;l5;l6];;

aplatir l7;;
(*[1;2;3;1;2;3;4;5;6;7;8;1;3;5;2;4;6;1;2;3;3;2;3]*)
```