TP7 INF201 Compte-rendu

(*Questions 1*)

```
type nat = int;; (*positive ou nul*)
type monome = int * nat;; (*int peut etre negatif aussi*)
type polynome = monome list;;
```

(*Question 2*)

```
(*a*)
let m1 : monome = (7,1);;
let m2 : monome = (-3,2);;
let m3 : monome = (1,4);;
let m4 : monome = (-1,4);;
let m5 : monome = (10,0);;

(*b*)
let p1 : polynome = [m5];;
let p2 : polynome = [m1;m5];;
let p3 : polynome = [m2;m1];;
let p4 : polynome = [m3;m2;m1];;
let p5 : polynome = [m4];;
```

(*Question 3*)

```
let deriveMono (kati:monome):monome=
let (a,b)=kati in let kapa=(a*b) in let kata=(b-1) in if kata<0 then (0,0) else (kapa,kata);;
```

(*Question 4*)

```
let rec derivPoly (pol:polynome):polynome=
match pol with
| [] -> []
| pr::fin -> [deriveMono pr] @ derivPoly fin (*Ca marche aussi en utlisant :: au lieu de @ si on
suprime les croches [] pare deriveMono pr*)
;;
```

(*Question 5*)

```
# assert (derivPoly [m1] = [(7,0)]);;
- : unit = ()
# assert (derivPoly [m2] = [(-6,1)]);;
- : unit = ()
# assert (derivPoly [m3] = [(4,3)]);;
- : unit = ()
# assert (derivPoly [m4] = [(-4,3)]);;
- : unit = ()
# assert (derivPoly [m5] = [(0,0)]);;
- : unit = ()

# assert (derivPoly p2 = [(7,0);(0,0)]);;
```

```
- : unit = ()
# assert (derivPoly p3 = [(-6,1);(7,0)]);;
- : unit = ()
# assert (derivPoly p4 = [(4,3);(-6,1)]);;
- : unit = ()
```

(*Question 6*)

```
let rec sommePoly_2 (k1:polynome)(k2:polynome):polynome=
match k1,k2 with
| [],[] -> []
| p,[] -> p
| [],m -> m
| pr1::fin1,pr2::fin2 -> let (a1,b1)=pr1 in let (a2,b2)=pr2 in
if b1=b2 then if a1+a2=0 then
sommePoly_2 fin1 fin2 else (a1+a2,b1)::sommePoly_2 fin1 fin2 else
if b1>b2 then (a1,b1)::sommePoly_2 fin1 fin2 else (a2,b2)::sommePoly_2 fin1 fin2;;

let rec sommePoly (k1:polynome)(k2:polynome):polynome=
match k1,k2 with
| [],[] -> []
| p,[] -> p
| [],m -> m
| (a1,b1)::fin1,(a2,b2)::fin2 ->
if b1=b2 then if a1+a2=0 then
sommePoly fin1 fin2 else (a1+a2,b1)::sommePoly fin1 fin2 else
if b1>b2 then (a1,b1)::sommePoly fin1 fin2 else (a2,b2)::sommePoly fin1 fin2;;
```

(*Question 7*)

```
# assert (sommePoly p4 p5 = sommePoly p5 p4 && sommePoly p4 p5 = p3);;
- : unit = ()
```

(*Question 8*)

```
# assert (derivPoly p3 = sommePoly([deriveMono (-3,2)] [deriveMono (7,1)]));;
- : unit = ()
```