Programmation Déclarative L3 MIAGE Examen Session1 Mars23
Eléments de Correction

April 28, 2023

Eléments de solution.

1 Somme des sommes

[1]:
```
%%writefile _tmp/progSomme.pl

sommesSuccessives([_E],[]).
sommesSuccessives([E,F|L],[D|R]):-
  {D=F+E},
  sommesSuccessives([F|L],R).

sommeSommes([E],[E]).
sommeSommes(L,R):-
  sommesSuccessives(L,T),
  sommeSommes(T,R).

main :-
  use_module(library(clpq)),
  sommesSuccessives([0, 4, 2, 0, 2, 3],R),
  writeln("Sommes successives :"),
  writeln(R),
  sommeSommes([0, 4, 2, 0, 2, 3],S),
  writeln("Somme des sommes :"),
  writeln(S).

:- main, halt.
```

```
Writing _tmp/progSomme.pl
```

[2]:
```
! swipl -q -s _tmp/progSomme.pl
```

```
Sommes successives :
[4,6,2,2,5]
Somme des sommes :
[53]
```

2 Alternance

[3]:
```
%%writefile _tmp/progAlternance.pl

separe([],[],[]).
separe([E|L],[E|P],N):-
  {E>0},
  separe(L,P,N).
separe([E|L],P,[E|N]):-
  {E<0},
  separe(L,P,N).
separe([0|L],[0|P],N):-
  separe(L,P,N).
separe([0|L],P,[0|N]):-
  separe(L,P,N).

fusionne([],[],[]).
fusionne([P],[],[P]).
fusionne([P|LP],[N|LN],[P,N|R]):-
    fusionne(LP,LN,R).


alternance(L,R):-
  separe(L,P,N),
  fusionne(P,N,R).
alternance(L,R):-
  separe(L,P,N),
  fusionne(N,P,R).
alternance(_L,[]).


main :-
  use_module(library(clpq)),
  alternance([-4, 2, 2, -3],S),
  writeln(S).

:- main, halt.
```

```
Writing _tmp/progAlternance.pl
```

[4]:
```
! swipl -q -s _tmp/progAlternance.pl
```

```
[2,-4,2,-3]
```

3 Racine carrée

[5]:
```
%%writefile _tmp/progSqrt.erl
-module(progSqrt).
-compile([export_all,nowarn_export_all]).

racineCarree(N) -> rc(N,0).

rc(N,K) when K*K > N -> K-1;
rc(N,K) -> rc(N,K+1).

main([]) ->
  io:format("~p",[racineCarree(13)]).
```

```
Writing _tmp/progSqrt.erl
```

[6]:
```
! escript _tmp/progSqrt.erl
```

```
3
```

Avec 2 processus :

[7]:
```
%%writefile progSqrt2.erl
-module(progSqrt2).
-compile([export_all,nowarn_export_all]).

racineCarree(N) ->
  spawn(progSqrt2,rcd,[N,0,self()]),
  spawn(progSqrt2,rcd,[N,1,self()]),
  receive R1 ->
    receive R2 when R1 < R2 -> R1;
            R2 -> R2 end end.

rcd(N,K,P) when K*K > N -> P!K-1;
rcd(N,K,P) -> rcd(N,K+2,P).

main([]) ->
  compile:file(progSqrt2),
  io:format("~p",[racineCarree(121)]).
```

```
Writing progSqrt2.erl
```

[8]:
```
! escript progSqrt2.erl
```

```
11
```

Avec N processus (chaque processus n'a plus besoin de faire une boucle, soit il a une réponse juste, soit il répond faux)

[9]:
```
%%writefile progSqrtN.erl
-module(progSqrtN).
-compile([export_all,nowarn_export_all]).

racineCarree(N) ->
  prods(N,N),
  consos(N,N).

prods(0,_N) -> done;
prods(K,N) ->
  spawn(progSqrtN,rcn,[N,K,self()]),
  prods(K-1,N).

consos(0,R) -> R;
consos(K,R) ->
  receive R1 when R1 < R -> consos(K-1,R1);
          _R1 -> consos(K-1,R) end.

rcn(N,K,P) when K*K > N -> P!K-1;
rcn(N,_K,P) -> P!N.

main([]) ->
  compile:file(progSqrtN),
  io:format("~p",[racineCarree(110)]).
```

```
Writing progSqrtN.erl
```

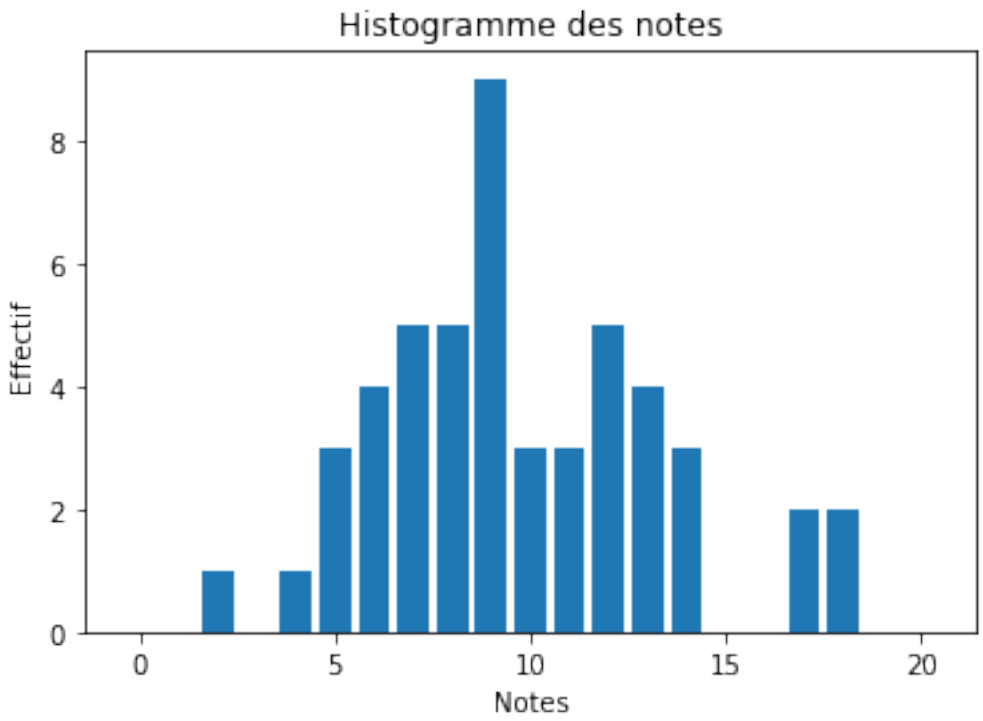
[10]:
```
! escript progSqrtN.erl
```

```
10
```

4 Notes

[11]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.
  ↪bar([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],[0,0,1,0,1,3,4,5,5,9,3,3,5,4,3,0
plt.xlabel("Notes")
plt.ylabel("Effectif")
plt.title("Histogramme des notes")
plt.show()
```



Fin.