Master Mind - corrigé

Guillaume Vernade - guillaume.vernade@ens.fr

Exercice 1

a.

```
aleatoire := () -> seq( rand(6)() +1,i=1..4);
```

b.

- Idée : on va créer 2 compteurs noirs et blancs qu'on incrémentera en fonction des égualités.
- Problème : comment traiter les cas ou plusieurs variables ont la même valeur (par ex. ne pas obtenir 4 noirs et 12 blancs lorsque a,b,c,d,A,B,C,D valent tous la même chose)
- Solution : se souvenir des variables "utilisées"
- Idée 1 : créer de nouvelles variables qui indiquent l'état des variables d'entrée
- Idée 2 : changer la valeur des variables de façon à ce qu'aucune égualité ne soit possible
- Problème : on ne peut pas changer la valeur des variables d'entrée
- Idée : on va en créer des copie dont on pourra change la valeur (e,f,g,h,E,F,G,H par ex.)
- Amélioration : en utilisant des listes on peut même faire des boucles.

```
noirs_blancs := proc (a,b,c,d, A,B,C,D)
  local test,sol,noirs,blancs,i,j;
  noirs := 0;
  blancs := 0;
  test := a,b,c,d;
  sol := A,B,C,D;
  for i from 1 to 4 do
    if test[i]=sol[i] then
      test[i]:=-1;
      sol[i]:=-2; # Des valeurs différentes pour éviter les égualités
      noirs:=noirs+1;
    fi;
  od;
  for i from 1 to 4 do
    for j from 1 to 4 do
      if i<>j and test[i]=sol[j] then
        test[i]:=-1;
        test[j]:=-2;
        blancs:=blancs+1;
      fi;
    od;
  od;
  noirs,blancs;
```

```
end;
```

c.

Le programme donné est pratiquement bon à quelques erreurs de type près...

Exercice 2

a.

Il s'agit juste de donne le quadruplet suivant dans $(Z/6Z)^4$ pour l'ordre lexicographique (on augmente le chiffre des unités de 1 puis on vérifie s'il ne vaut pas 7, auquel cas on le remplace par un 1 et on augmente le chiffre suivant, et on recommence).

```
coup_suivant := proc(a,b,c,d)
  local X,i;
  X := [a,b,c,d+1];
  for i from 4 to 2 by -1 do
    if X[i]=7 then
      X[i]:=1;
      X[i-1]:=X[i-1]+1;
    fi;
  od;
  op(X);
end;
```

b.

On teste si la combinaison donne le bon nombre de noirs et de blancs que les tests précédents. Vu la façon que l'on a choisi pour stocker la partie (quelque chose de récursif (partie=[a,b,c,d,noirs,blancs,partie])) on va faire un programme récursif.

```
correcte := proc(a,b,c,d, partie)
  local partie_suiv,A,B,C,D,noirs,blancs;
  if partie=[] then
    true; #Si on a pas de test à faire, la combinaison est bonne (c'est le cas d'arret)
  else
    A:=partie[1];
    B:=partie[2];
    C:=partie[3];
    D:=partie[4];
    noirs:=partie[5];
    blancs:=partie[6];
    partie_suiv:=partie[7];
    if noirs,blancs=noirs_blancs(a,b,c,d,A,B,C,D) then
      correcte(a,b,c,d,partie_suiv); #Si la combinaison donne le bon résultat on teste la suite
    else
      false; #Sinon pas besoin d'aller plus loin
    fi;
  fi;
end;
```

c.

On commence par tester la première combinaison possible (1 1 1 1) puis on teste les combinaisons dans l'ordre (avec *suivant*), jusqu'à en trouver une correcte (on teste avec *correcte*), pour trouver le coup suivant. On continue jusqu'à trouver la solution.

```
resolution := proc(A,B,C,D)
  local a,b,c,d,noirs,blancs,partie,gagne,ligne;
  a,b,c,d := 1,1,1,1;
  while ligne<10 and not gagne do
    noirs,blancs:=noirs_blancs(a,b,c,d,A,B,C,D); # Ou on demande au joueur
    printf("Je joue %d %d %d %d, j'obtiens %d bons et %d mal placés\n",a,b,c,d,noirs,blancs);
    if noirs=4 then
      gagne=true
    else
      partie:=[a,b,c,d,noirs,blancs,partie];
      while a<>7 and not correcte(a,b,c,d,partie) do
a,b,c,d:=coup_suivant(a,b,c,d);
      od;
      if a=7 then
        printf("Il n'y a pas de solution\n");
gagne:=true;
      fi;
    fi;
    ligne:=ligne+1;
  od;
end;
```

Exercice 3

a.

On reprend le même algo[1] sans les *printf* (ça servira à la question suivante pour ne pas avoir trop de trucs affichés). Il renvoie déjà le nombre de coups joués.

b.

On teste le nombre de coups pour chaque combinaison.

```
stat := proc()
  local nb_coups,a,b,c,d,statistiques;
  statistiques:=[0\$10];
  a,b,c,d:=1,1,1,1;
  while a<>7 do
    nb_coups:=resolution(a,b,c,d);
    statistiques[nb_coups]:=statistiques[nb_coups]+1;
    a,b,c,d:=coup_suivant(a,b,c,d);
  od;
```

[1] En fait si on avait demandé au joueur le résultat en pion noirs et blancs il aurait fallut remplacer cette partie par celle que j'ai notée au dessus

```
  statistiques;
end;
```

c.

À priori une question de plus au maximum par couleur suplémentaire.