Projet: Échecs

Simon BENSIAM
Asif AHMED SHERAZI

3 décembre 2018

Mode d'emploi

Notre programme permet de saisir et de résoudre des problèmes d'échecs. Au début le plateau sera vide (comme ci-dessous). Afin de faire fonctionner correctement le programme, quand on l'exécute il nous demande si l'on souhaite entrer des pièces. Par la suite le programme souhaite connaitre l'endroit ou nous voulons placer la pièce ainsi que son type. Par exemple pour poser un roi noir il faut écrire "Rn" puis la case ou l'on souhaite le poser et ainsi de suite pour chaque pièce supplémentaire. (Rb et Rn = Roi blanc et Roi noir) (Db et Dn = Dame blanche et Dame noire) (Tb et Tn = Tour blanche et Tour noire) (Fb et Fn = Fou blanc et Fou noir) (Cb et Cn = Cavalier blanc et Cavalier noir) (Pb et Pn = Pion blanc et Pion noir)

Quand le plateau est vide toutes ses cases sont initialisées à -1.

	A	B	C	D	E	F	G	H	
8	-1	-1	-1	-1	-1	-1	-1	-1	8
7	-1	-1	-1	-1	-1	-1	-1	-1	7
6	-1	-1	-1	-1	-1	-1	-1	-1	6
5	-1	-1	-1	-1	-1	-1	-1	-1	5
4	-1	-1	-1	-1	-1	-1	-1	-1	4
3	-1	-1	-1	-1	-1	-1	-1	-1	3
2	-1	-1	-1	-1	-1	-1	-1	-1	2
1	-1	-1	-1	-1	-1	-1	-1	-1	1
	A	B	C	D	E	F	G	H	

Une fois que l'on a fini de saisir toutes les pièces, il suffit de répondre non "n" à cette question : Voulez-vous placez une nouvelle pièce ? (o/n).
Ensuite le programme vous demandera qui doit jouer, 0 correspond aux blancs et 1 aux noirs. Puis il faudra choisir en combien de coups l'échec et mat doit être réalisé. Le programme fait ensuite une recherche parmis tout les scénarios possibles et détermine si l'échec et mat est réalisable.

Exemple d'utilisation

Nous allons prendre un exemple simple avec un échec et mat en 1 coup. Quand le programme s'exécute il commence ainsi :

	A	B	C	D	E	F	G	H	
8									8
7									7
6									6
5									5
4									4
3									3
2									2
1									1
	A	B	C	D	E	F	G	H	

Voulez-vous placez une nouvelle pièce ? (o/n)
o
Ou voulez-vous placez votre pièce ? (Exemple : A1)
A8
Quel type de pièce ? (Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)
rn

	A	B	C	D	E	F	G	H	
8	Rn								8
7									7
6									6
5									5
4									4
3									3
2									2
1									1
	A	B	C	D	E	F	G	H	

Le roi noir a été placé, continuons avec les autres pièces.
Voulez-vous placez une nouvelle pièce ? (o/n)
o
Ou voulez-vous placez votre pièce ? (Exemple : A1)
C8
Quel type de pièce ? (Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)
fb

	A	B	C	D	E	F	G	H	
8	Rn		Fb						8
7									7
6									6
5									5
4									4
3									3
2									2
1									1
	A	B	C	D	E	F	G	H	

Voulez-vous placez une nouvelle pièce ? (o/n)
o
Ou voulez-vous placez votre pièce ? (Exemple : A1)
f7
Quel type de pièce ? (Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)
db

	A	B	C	D	E	F	G	H	
8	Rn		Fb						8
7						Db			7
6									6
5									5
4									4
3									3
2									2
1									1
	A	B	C	D	E	F	G	H	

Voulez-vous placez une nouvelle pièce ? (o/n)
o
Ou voulez-vous placez votre pièce ? (Exemple : A1)
e1
Quel type de pièce ? (Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)
rb

	A	B	C	D	E	F	G	H	
8	Rn		Fb						8
7						Db			7
6									6
5									5
4									4
3									3
2									2
1					Rb				1
	A	B	C	D	E	F	G	H	

Voulez-vous placez une nouvelle pièce ? (o/n)
n
Maintenant que nous avons placé toutes les pièces souhaitées on l'arrête pour passer à la résolution du problème :
Qui doit jouer ? (0 pour blanc / 1 pour noir)
0
Echec et mat en combien de coups ?
1
Recherche en cours : 2% de scénarios testés
Recherche en cours : 5% de scénarios testés
Recherche en cours : 8% de scénarios testés
Recherche en cours : 11% de scénarios testés
Recherche en cours : 14% de scénarios testés
Recherche en cours : 17% de scénarios testés
Recherche en cours : 20% de scénarios testés
Recherche en cours : 22% de scénarios testés
Recherche en cours : 25% de scénarios testés
Recherche en cours : 28% de scénarios testés
Recherche en cours : 31% de scénarios testés
Recherche en cours : 34% de scénarios testés
Recherche en cours : 37% de scénarios testés
coup 1 : F7 va en B7
Echec et mat !!!

Exemple d'utilisation numéro 2

Ici nous allons faire un échec et mat en 3 coups en faisant jouer les blancs.
On va supposer que toutes les pièces ont déjà été posées.

	A	B	C	D	E	F	G	H	
8		Cn					Rn		8
7	Tn						Cn	Pn	7
6		Pn					Pb	Fb	6
5	Pn		Pn						5
4							Pb		4
3			Pb					Pb	3
2	Dn				Fb				2
1					Db	Tb	Rb		1
	A	B	C	D	E	F	G	H	

coup 1 : E2 va en C4
coup 2 : G8 va en H8
coup 3 : E1 va en E8
coup 4 : G7 va en E8
coup 5 : F1 va en F8
Echec et mat !!!
OU
coup 3 : F1 va en F8
Echec et mat !!!
OU
coup 2 : A7 va en F7
coup 3 : G6 va en F7
coup 4 : G8 va en F8
coup 5 : E1 va en E8
Echec et mat !!!
OU
coup 4 : G8 va en H8
coup 5 : F7 va en F8

Echec et mat !!!

OU

coup 2 : G7 va en E6
coup 3 : E1 va en E6
coup 4 : G8 va en H8
coup 5 : E6 va en E8

Echec et mat !!!

OU

coup 4 : A7 va en F7
coup 5 : E6 va en E8

Echec et mat !!!

OU

coup 3 : F1 va en F8

Echec et mat !!!

OU

coup 2 : A2 va en C4
coup 3 : E1 va en E8
coup 4 : G7 va en E8
coup 5 : F1 va en F8

Echec et mat !!!

Les fonctions

— **struct plateau :** contient 1 tableau à deux dimensions d'entiers qui correspond à l'échiquier et un second tableau d'entier à une dimension qui contient le nombre de pièce de chaque type posé sur l'échiquier.
— **struct solution :** contient un entier (0 ou 1) qui correspond a si l'échec et mat est possible ainsi que l'historique des coups pour l'atteindre sous forme d'arbre.
— **void initp :** prend en entrée un pointeur sur un plateau et met toutes les cases à -1 (vide).
— **void newp :** prend un pointeur sur un tableau en paramètre et permet de saisir un nouveau problème.
— **void affp :** prend en entrée un plateau et l'affiche.
— **void affc :** affiche chaque case du plateau.
— **int convert :** En fonction du type de pièce entrée, renvoie l'entier correspondant à ce type (ex : Rn = 1).
— **void viderBuffer :** permet de vider l'entrée clavier.
— **void deplp :** prend en entrée deux coordonnées et déplace la pièce qui se trouve à la première coordonnée vers la seconde, effectue aussi la promotion des pions.
— **void decomptepieces :** actualise le champ nbp de la structure plateau.
— **int tropdepieces :** renvoie 1 si il y a trop de pièce d'un certain type sur le plateau (ex : 2 Rois blanc).
— **void coupslegaux :** renvoie tous les coups légaux d'une pièce qui se trouve à la coordonnée passée en paramètre.
— **void ajoutercoup :** prend un tableau d'entier en paramètre et ajoute dans celui-ci les coups légaux renvoyés par la fonction coupslégaux.
— **void initlscoup :** écrit -1 dans toutes les cases de la liste de coups qui sera envoyée à la fonction coupslégaux.
— **int coordvalide :** renvoie 1 si la coordonée en paramètre est une des coordonnées d'une des cases du plateau.
— **char itoc :** prend un entier (0 à 7) et renvoie le caractère correspondant (A à H).
— **int echec :** renvoie 1 si le roi est en échec, 0 sinon.
— **int mat :** renvoie 1 si il y a echec et mat, 0 sinon.
— **void selecp :** permet de sélectionner le type de problème.
— **solution resop :** recherche si l'échec et mat est possible (fonction principale).
— **arbre initarbre, void inserefeuille, void inserearbre :** stocke l'historique des coups à faire pour faire échec et mat.
— **void afficherarbre :** affiche la solution à la fin du programme si il y en a une, affiche le champ hist de la structure solution qui est renvoyée à la fin du programme.

Header

```c
#include <stdio.h>
#include <stdlib.h>

#define couleur(param) printf("\033[%sm",param) //Changer la couleur des pièces

typedef struct plateau{
        int grille[8][8];
        int Nbp[12];
}plateau;

struct noeud {
        int coord;       //Coordonnée de départ
        int coup;        //Coordonnée d'arrivée
        struct noeud * suiv[];
};
typedef struct noeud  noeud;
typedef struct noeud * arbre;

typedef struct solution{
        int sol;
        arbre hist;
}solution;

void initp (plateau *p); //initialiser le plateau avec les pièces
```

```
25
26 void newp(plateau *p); //Créer un nouveau problème
27
28 void affp (plateau p); //Afficher le plateau de jeu
29
30 void affc (int c); //Afficher une case du plateau
31
32 int convert(char * nom); //Convertir le nom de la pièce en entier
33
34 void viderBuffer(); //Vider stdin
35
36 void deplp(plateau *p, int dep, int arr);
37
38 void decomptepieces(plateau *p);
39
40 int tropdepieces(plateau p, int t);
41
42 void coupslegaux(plateau p, int coord, int lscoup[27], int mode);
43
44 void ajoutercoup(int lscoup[27], int coord);
45
46 void initlscoup(int lscoup[27]);
47
48 void afflscoup(int lscoup[27]);
49
50 int coordvalide(int coord);
51
52 char itoc(int n);
53
54 int echec(plateau p, int c);
55
56 int mat(plateau p, int couleur);
57
58 void selecp(plateau p);
```

```
59
60 solution resop(plateau p, int coup, int nbc, int c);
61
62 char converttype(int type);
63
64 void inserefeuille (arbre a,int coord, int c);
65
66 arbre initarbre();
67
68 void inserearbre (arbre a, arbre b, int coord, int c);
69
70 void afficherarbre(arbre a, int nbc);
```

Listing du programme

```
1  #include "echecs.h"
2
3  int main(){
4          plateau p;
5          initp(&p);
6          newp(&p);
7          affp(p);
8          selecp(p);
9          return 0;
10 }
11
12 void selecp(plateau p){
13         char c;
14         printf("Qui doit jouer ?   (0 pour blanc / 1 pour noir)\n");
15         scanf("%c",&c);
16         viderBuffer();
17         while(c != '0' && c != '1'){
18                 printf("Réponse invalide (0/1)\n");
19                 scanf("%c",&c);
20                 viderBuffer();
21         }
22         c -= '0';
23
24         char nbc;
```

```
25          printf("Echec et mat en combien de coups ?\n");
26          scanf("%c",&nbc);
27          viderBuffer();
28          while(nbc < '1'){
29                  printf("Réponse invalide\n");
30                  scanf("%c",&nbc);
31                  viderBuffer();
32          }
33          nbc -= '0';
34
35          solution s = resop(p, 1, nbc*2 -1, c);
36
37          if(s.sol){
38                  printf("\n");
39                  affp(p);
40                  afficherarbre(s.hist, 1);
41          }
42          else
43                  printf("Pas de solution...\n");
44  }
45
46  solution resop(plateau p, int coup, int nbc , int c){
47          int lscoup[27];
48          int i,j;
49          int k;
50          plateau p2;
51          int z = 1;
52
53          solution solutionact;
54          solutionact.hist = initarbre();
55
56          solution solutionsuiv;
57          solutionsuiv.hist = initarbre();
58
```

```
59          int nc = 0;
60          if (coup == 1){
61                  for(i=0 ; i<8 ; i++){
62                          for(j=0; j<8 ; j++){
63                                  if(p.grille[i][j] != -1 && p.grille[i][j]%2 == c){
64                                          initlscoup(lscoup);
65                                          coupslegaux(p, i*10 + j, lscoup, 1);
66                                          for(k=0 ; lscoup[k] != -1 ; k++)
67                                                  nc++;
68                                  }
69                          }
70                  }
71          }
72          p2 = p;
73          if(coup%2 == 1){
74                  solutionact.sol = 0;
75                  if(coup <= nbc){
76                          for(i=0 ; i<8 ; i++){
77                                  for(j=0 ; j<8 ; j++){
78                                          if(p.grille[i][j] != -1 && p.grille[i][j]%2 == c){
79                                                  initlscoup(lscoup);
80                                                  coupslegaux(p2, i*10 + j, lscoup, 1);
81                                                  for(k=0 ; lscoup[k] != -1 ; k++){
82                                                          deplp(&p2, i*10 + j, lscoup[k]);
83                                                          if(echec(p2,c ? 0:1)){
84                                                                  if(mat(p2,c ? 0:1)){
85                                                                          solutionact.sol = 1;
86                                                                          inserefeuille(solutionact.
                                                                             hist, i*10 + j, lscoup[k
                                                                             ]);
87                                                                          return solutionact; //Mettre
                                                                             en commentaire cette
                                                                             ligne pour obtenir toutes
                                                                             les solutions et pas
```

```
                                                                    seulement la 1ere trouvée
88                                                                          }
89                                                                  }
90                                                                  solutionsuiv = resop(p2, coup + 1, nbc, c ?
                                                                        0:1);
91                                                                  if(solutionsuiv.sol){
92                                                                          inserearbre(solutionact.hist,
                                                                            solutionsuiv.hist, i*10 + j,
                                                                            lscoup[k]);
93                                                                          solutionact.sol = 1;
94                                                                  }
95                                                                  p2 = p;
96                                                                  if(coup == 1){
97                                                                          printf("Recherche en cours : %d%c de
                                                                            scénarios testés\n", z++*100/nc,
                                                                             '%');
98                                                                  }
99                                                          }
100                                                 }
101                                         }
102                                 }
103                                 return solutionact;
104                         }
105                         solutionact.sol = 0;
106                 }
107                 else{
108                         if(coup <= nbc){
109                                 solutionact.sol = 1;
110                                 for(i=0 ; i<8 ; i++){
111                                         for(j=0 ; j<8 ; j++){
112                                                 if(p.grille[i][j] != -1 && p.grille[i][j]%2 == c){
113                                                         initlscoup(lscoup);
114                                                         coupslegaux(p2, i*10 + j, lscoup, 1);
115                                                         for(k=0 ; lscoup[k] != -1 ; k++){
```

```
116                                                     deplp(&p2, i*10 + j, lscoup[k]);
117                                                     solutionsuiv = resop(p2, coup + 1, nbc, c ?
                                                            0:1);
118                                                     solutionact.sol *= solutionsuiv.sol;
119                                                     if(solutionsuiv.sol)
120                                                             inserearbre(solutionact.hist,
                                                                    solutionsuiv.hist, i*10 + j,
                                                                    lscoup[k]);
121                                                     p2 = p;
122                                             }
123                                     }
124                             }
125                     }
126                     if(!solutionact.sol)
127                             free(solutionact.hist);
128                     return solutionact;
129             }
130             solutionact.sol = 0;
131     }
132     return solutionact;
133 }
134
135 void initp(plateau *p){
136     int i,j;
137     for(i=0 ; i<8 ; i++){
138             for(j=0 ; j<8 ; j++){
139                     p->grille[i][j] = -1;
140             }
141     }
142     for(i=0 ; i<13 ; i++){
143             p->Nbp[i] = 0;
144     }
145     decomptepieces(p);
146 }
```

```
147
148 void newp(plateau *p){
149         int r = 1;
150         char r1;
151         char r2;
152         char coord[2];
153         char type[2];
154         int o,a; //Ordonnée et abscisse
155         int t; //Type de pièce
156         while(r){
157                 affp(*p);
158                 printf("Voulez-vous placez une nouvelle pièce ? (o/n)\n");
159                 scanf("%c",&r1);
160                 viderBuffer();
161                 while(r1 != 'o' && r1 != 'n'){
162                         printf("Réponse invalide (o/n)\n");
163                         scanf("%c",&r1);
164                         viderBuffer();
165                 }
166                 if (r1 == 'o'){
167                         printf("Ou voulez-vous placez votre pièce ? (Exemple : A1)\n");
168                         scanf("%s",coord);
169                         viderBuffer();
170                         while(coord[0] < 'A' || coord[0] > 'H' || coord[1] < '1' || coord[1] > '8'){
171                                 if (coord[0] >= 'a' && coord[0] <= 'h'){
172                                         coord[0] += 'A' - 'a';
173                                         break;
174                                 }
175                                 printf("Les coordonnées que vous avez entré sont incorrectes. Ex:[A
                                    à H][1 à 8]\nVeuillez saisir de nouvelles coordonées\n");
176                                 scanf("%s",coord);
177                                 viderBuffer();
178                         }
179                         a = coord[0] - 'A';
```

```
180                 o = (coord[1] - '8') * -1;
181                 if (p->grille[o][a] != -1){
182                         printf("Il y a déjà une pièce sur cette case voulez-vous la
                               remplacer ? (o/n)\n");
183                         scanf("%c",&r2);
184                         viderBuffer();
185                         while(r2 != 'o' && r2 != 'n'){
186                                 printf("Réponse invalide (o/n)\n");
187                                 scanf("%c",&r2);
188                                 viderBuffer();
189                         }
190                         if(r2 == 'n'){
191                                 continue;
192                         }
193                         p->grille[o][a] = -1;
194                         decomptepieces(p);
195                 }
196                 printf("Quel type de pièce ? (Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn /
                        Cb / Pn / Pb)\n");
197                 scanf("%s",type);
198                 viderBuffer();
199                 while(type[0] != 'R' && type[0] != 'D' && type[0] != 'T' && type[0] != 'F'
                        && type[0] != 'C' && type[0] != 'P'){
200                         if(type[0] == 'r' || type[0] == 'd' || type[0] == 't' || type[0] ==
                               'f' || type[0] == 'c' || type[0] == 'p'){
201                                 type[0] += 'A' - 'a';
202                                 break;
203                         }
204                         printf("Le type de pièce que vous avez entré est incorrect.\nEx:(Rn
                               / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)\nVeuillez
                               saisir à nouveau le type de pièce\n");
205                         scanf("%s",type);
206                         viderBuffer();
207                 }
```

```
208                 while(type[1] != 'n' && type[1] != 'b'){
209                         printf("La couleur de la pièce est incorrecte. (n pour noir / b pour
                              blanc)\nVeuillez saisir une nouvelle couleur\n");
210                         char bn;
211                         scanf("%c",&bn);
212                         viderBuffer();
213                         type[1] = bn;
214                 }
215                 t = convert(type);
216
217
218                 if (tropdepieces(*p,t)){
219                         printf("Il y a trop de pièces de ce type !!!\n\n");
220                         continue;
221                 }
222
223                 if (t == 10 || t == 11){
224                         if(o == 0 || o == 7){
225                                 printf("Il ne peut pas y avoir de pion ici !!!\n\n");
226                                 continue;
227                         }
228                 }
229                 if (t == 0){
230                         if(     (p->grille[o - 1][a]     == 1    && coordvalide((o * 10 + a)
                                    - 10))      ||
231                                 (p->grille[o - 1][a - 1] == 1    && coordvalide((o * 10 + a)
                                    - 11))      ||
232                                 (p->grille[o - 1][a + 1] == 1    && coordvalide((o * 10 + a)
                                    - 9))       ||
233                                 (p->grille[o][a - 1]     == 1    && coordvalide((o * 10 + a)
                                    - 1))       ||
234                                 (p->grille[o][a + 1]     == 1    && coordvalide((o * 10 + a)
                                    + 1))       ||
```

```
235                         (p->grille[o + 1][a - 1] == 1   && coordvalide((o * 10 + a)
                               + 9))        ||
236                         (p->grille[o + 1][a + 1] == 1   && coordvalide((o * 10 + a)
                               + 11))       ||
237                         (p->grille[o + 1][a]     == 1   && coordvalide((o * 10 + a)
                               + 10))) {
238
239                         printf("Il ne peut pas y avoir de roi blanc ici !!!\n\n");
240                         continue;
241                 }
242         }
243         if (t == 1){
244                 if(     (p->grille[o - 1][a]     == 0   && coordvalide((o * 10 + a)
                           - 10))       ||
245                         (p->grille[o - 1][a - 1] == 0   && coordvalide((o * 10 + a)
                               - 11))       ||
246                         (p->grille[o - 1][a + 1] == 0   && coordvalide((o * 10 + a)
                               - 9))        ||
247                         (p->grille[o][a - 1]     == 0   && coordvalide((o * 10 + a)
                               - 1))        ||
248                         (p->grille[o][a + 1]     == 0   && coordvalide((o * 10 + a)
                               + 1))        ||
249                         (p->grille[o + 1][a - 1] == 0   && coordvalide((o * 10 + a)
                               + 9))        ||
250                         (p->grille[o + 1][a + 1] == 0   && coordvalide((o * 10 + a)
                               + 11))       ||
251                         (p->grille[o + 1][a]     == 0   && coordvalide((o * 10 + a)
                               + 10)))      {
252
253                         printf("Il ne peut pas y avoir de roi noir ici !!!\n\n");
254                         continue;
255                 }
256         }
257
```

```
258                         p->grille[o][a] = t;
259                         decomptepieces(p);
260                 }
261                 else{
262                         if(p->Nbp[0] == 0 || p->Nbp[1] == 0){
263                                 printf("Il manque au moins un roi sur l'échiquier !!!\n\n");
264                                 continue;
265                         }
266                         r = 0;
267                 }
268         }
269 }
270
271 int tropdepieces(plateau p, int t){
272         int promotion = 0, i;
273         for(i=2 ; i<10 ; i++){
274                 if(i == 2 || i == 3){
275                         if(p.Nbp[i]>1)
276                                 promotion += p.Nbp[i] - 1;
277                 }
278                 else if(p.Nbp[i]>2)
279                         promotion += p.Nbp[i] - 2;
280         }
281         switch (t){
282                 case 0:
283                 if(p.Nbp[0] == 1)
284                         return 1;
285                 break;
286
287                 case 1:
288                 if(p.Nbp[1] == 1)
289                         return 1;
290                 break;
291
```

```
292                 case 2:
293                 if(p.Nbp[2] > 1){
294                         if(1 + promotion + p.Nbp[10] == 9)
295                                 return 1;
296                         break;
297                 }
298                 if(p.Nbp[2] + promotion + p.Nbp[10] == 9)
299                         return 1;
300                 break;
301
302                 case 3:
303                 if(p.Nbp[3] > 1){
304                         if(1 + promotion + p.Nbp[11] == 9)
305                                 return 1;
306                         break;
307                 }
308                 if(p.Nbp[3] + promotion + p.Nbp[11] == 9)
309                         return 1;
310                 break;
311
312                 case 4:
313                 if(p.Nbp[4] > 2){
314                         if(2 + promotion + p.Nbp[10] == 10)
315                                 return 1;
316                         break;
317                 }
318                 if(p.Nbp[4] + promotion + p.Nbp[10] == 10)
319                         return 1;
320                 break;
321
322                 case 5:
323                 if(p.Nbp[5] > 2){
324                         if(2 + promotion + p.Nbp[11] == 10)
325                                 return 1;
```

```
326                 break;
327             }
328             if(p.Nbp[5] + promotion + p.Nbp[11] == 10)
329                 return 1;
330             break;
331
332             case 6:
333             if(p.Nbp[6] > 2){
334                 if(2 + promotion + p.Nbp[10] == 10)
335                     return 1;
336                 break;
337             }
338             if(p.Nbp[6] + promotion + p.Nbp[10] == 10)
339                 return 1;
340             break;
341
342             case 7:
343             if(p.Nbp[7] > 2){
344                 if(2 + promotion + p.Nbp[11] == 10)
345                     return 1;
346                 break;
347             }
348             if(p.Nbp[7] + promotion + p.Nbp[11] == 10)
349                 return 1;
350             break;
351
352             case 8:
353             if(p.Nbp[8] > 2){
354                 if(2 + promotion + p.Nbp[10] == 10)
355                     return 1;
356                 break;
357             }
358             if(p.Nbp[8] + promotion + p.Nbp[10] == 10)
359                 return 1;
```

```
360                     break;
361
362                     case 9:
363                     if(p.Nbp[9] > 2){
364                             if(2 + promotion + p.Nbp[11] == 10)
365                                     return 1;
366                             break;
367                     }
368                     if(p.Nbp[9] + promotion + p.Nbp[11] == 10)
369                             return 1;
370                     break;
371
372                     case 10:
373                     if(p.Nbp[10] + promotion == 8)
374                             return 1;
375                     break;
376
377                     case 11:
378                     if(p.Nbp[11] + promotion == 8)
379                             return 1;
380                     break;
381
382                     default:
383                     return 0;
384             }
385             return 0;
386 }
387
388 int convert(char * nom){
389             switch (nom[0]){
390                     case 'R':
391                     if(nom[1]=='b')
392                             return 0;
393                     return 1;
```

```
394                 break;
395
396                 case 'D':
397                 if(nom[1]=='b')
398                         return 2;
399                 return 3;
400                 break;
401
402                 case 'T':
403                 if(nom[1]=='b')
404                         return 4;
405                 return 5;
406                 break;
407
408                 case 'F':
409                 if(nom[1]=='b')
410                         return 6;
411                 return 7;
412                 break;
413
414                 case 'C':
415                 if(nom[1]=='b')
416                         return 8;
417                 return 9;
418                 break;
419
420                 case 'P':
421                 if(nom[1]=='b')
422                         return 10;
423                 return 11;
424                 break;
425
426                 default:
427                 return -1;
```

```
428         }
429 }
430
431 void affp(plateau p){
432         int i,j;
433         printf("␣␣|␣A␣|␣B␣|␣C␣|␣D␣|␣E␣|␣F␣|␣G␣|␣H␣|\n␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣+
                ␣␣+␣␣\n");
434         for(i=0 ; i<8 ; i++){
435                 printf("%d␣|",8-i);
436                 for(j=0 ; j<8 ; j++)
437                         affc(p.grille[i][j]);
438                 printf("␣%d",8-i);
439                 printf("\n␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣+␣␣\n");
440         }
441         printf("␣␣|␣A␣|␣B␣|␣C␣|␣D␣|␣E␣|␣F␣|␣G␣|␣H␣|\n\n");
442 }
443
444 void affc(int c){
445         switch (c){
446                 case -1:
447                 printf("␣␣␣|");
448                 break;
449
450                 case 0:
451                 printf("␣R␣|");
452                 break;
453
454                 case 1:
455                 couleur("31");
456                 printf("␣R␣");
457                 couleur("0");
458                 printf("|");
459                 break;
460
```

```
461                 case 2:
462                 printf("␣D␣|");
463                 break;
464 
465                 case 3:
466                 couleur("31");
467                 printf("␣D␣");
468                 couleur("0");
469                 printf("|");
470                 break;
471 
472                 case 4:
473                 printf("␣T␣|");
474                 break;
475 
476                 case 5:
477                 couleur("31");
478                 printf("␣T␣");
479                 couleur("0");
480                 printf("|");
481                 break;
482 
483                 case 6:
484                 printf("␣F␣|");
485                 break;
486 
487                 case 7:
488                 couleur("31");
489                 printf("␣F␣");
490                 couleur("0");
491                 printf("|");
492                 break;
493 
494                 case 8:
```

```
495                     printf("␣C␣|");
496                     break;
497 
498                     case 9:
499                     couleur("31");
500                     printf("␣C␣");
501                     couleur("0");
502                     printf("|");
503                     break;
504 
505                     case 10:
506                     printf("␣P␣|");
507                     break;
508 
509                     case 11:
510                     couleur("31");
511                     printf("␣P␣");
512                     couleur("0");
513                     printf("|");
514                     break;
515 
516                     default:
517                     printf("???|");
518             }
519 }
520 
521 void viderBuffer(){
522     int c;
523     do {
524         c = getchar();
525     } while (c != '\n' && c != EOF);
526 }
527 
528 void decomptepieces(plateau *p){
```

```
529         int i,j;
530         for(i=0 ; i<12 ; i++)
531                 p->Nbp[i] = 0;
532
533         for(i=0 ; i<8 ; i++){
534                 for(j=0 ; j<8 ; j++){
535                         if(p->grille[i][j] != -1)
536                                 p->Nbp[p->grille[i][j]] += 1;
537                 }
538         }
539 }
540
541 int coordvalide(int coord){
542         int i;
543         for(i=0 ; i<64 ; i++){
544                 if(     (coord >= 00 && coord <= 07) ||
545                         (coord >= 10 && coord <= 17) ||
546                         (coord >= 20 && coord <= 27) ||
547                         (coord >= 30 && coord <= 37) ||
548                         (coord >= 40 && coord <= 47) ||
549                         (coord >= 50 && coord <= 57) ||
550                         (coord >= 60 && coord <= 67) ||
551                         (coord >= 70 && coord <= 77))
552                         return 1;
553         }
554         return 0;
555 }
556
557 void deplp(plateau *p, int dep, int arr){
558         p->grille[arr/10][arr%10]=p->grille[dep/10][dep%10];
559         p->grille[dep/10][dep%10]= -1;
560
561         if (p->grille[arr/10][arr%10] == 10 && arr/10 == 0){ //promotion pion blanc
562                 p->grille[arr/10][arr%10] = 2;
```

```
563                     decomptepieces(p);
564             }
565
566             else if (p->grille[arr/10][arr%10] == 11 && arr/10 == 7){ //promotion pion noir
567                     p->grille[arr/10][arr%10] = 3;
568                     decomptepieces(p);
569             }
570 }
571
572 void coupslegaux(plateau p, int coord, int lscoup[27], int mode){
573             int type;
574             type = p.grille[coord/10][coord%10];
575             int o = coord/10;
576             int a = coord%10;
577             int i;
578             int c = 1;
579             int d = 0;
580             plateau p2;
581             p2 = p;
582
583             switch (type){
584                     case 0: //Roi Blanc
585                     c = 0;
586
587                     case 1: //Roi Noir
588                     if((p.grille[o - 1][a - 1])%2 != c && coordvalide(coord - 11)){
589                             deplp(&p2,coord,coord - 11);
590                             if(!echec(p2,c))
591                                     ajoutercoup(lscoup, coord - 11);
592                             p2 = p;
593                     }
594
595                     if((p.grille[o - 1][a])%2 != c && coordvalide(coord - 10)){
596                             deplp(&p2,coord,coord - 10);
```

```
597                     if(!echec(p2,c))
598                             ajoutercoup(lscoup, coord - 10);
599                     p2 = p;
600             }
601
602             if((p.grille[o - 1][a + 1])%2 != c && coordvalide(coord - 9)){
603                     deplp(&p2,coord,coord - 9);
604                     if(!echec(p2,c))
605                             ajoutercoup(lscoup, coord - 9);
606                     p2 = p;
607             }
608
609             if((p.grille[o][a - 1])%2 != c && coordvalide(coord - 1)){
610                     deplp(&p2,coord,coord - 1);
611                     if(!echec(p2,c))
612                             ajoutercoup(lscoup, coord - 1);
613                     p2 = p;
614             }
615
616             if((p.grille[o][a + 1])%2 != c && coordvalide(coord + 1)){
617                     deplp(&p2,coord,coord + 1);
618                     if(!echec(p2,c))
619                             ajoutercoup(lscoup, coord + 1);
620                     p2 = p;
621             }
622
623             if((p.grille[o + 1][a - 1])%2 != c && coordvalide(coord + 9)){
624                     deplp(&p2,coord,coord + 9);
625                     if(!echec(p2,c))
626                             ajoutercoup(lscoup, coord + 9);
627                     p2 = p;
628             }
629
630             if((p.grille[o + 1][a])%2 != c && coordvalide(coord + 10)){
```

```
631                     deplp(&p2,coord,coord + 10);
632                     if(!echec(p2,c))
633                             ajoutercoup(lscoup, coord + 10);
634                     p2 = p;
635             }
636
637             if((p.grille[o + 1][a + 1])%2 != c && coordvalide(coord + 11)){
638                     deplp(&p2,coord,coord + 11);
639                     if(!echec(p2,c))
640                             ajoutercoup(lscoup, coord + 11);
641                     p2 = p;
642             }
643             break;
644
645             case 2: //Dame Blanche
646             d = 1;
647
648             case 3: //Dame Noire
649             for(i=1 ; (p.grille[o - i][a])%2 == -1 && coordvalide(coord - i*10) ; i++){
650                     if(mode){
651                             deplp(&p2, coord, coord - i*10);
652                             if(!echec(p2,d ? 0 : 1))
653                                     ajoutercoup(lscoup, coord - i*10);
654                             p2 = p;
655                     }
656                     else
657                             ajoutercoup(lscoup, coord - i*10);
658             }
659             if((p.grille[o - i][a])%2 == d && coordvalide(coord - i*10)){
660                     if(mode){
661                             deplp(&p2, coord, coord - i*10);
662                             if(!echec(p2,d ? 0 : 1))
663                                     ajoutercoup(lscoup, coord - i*10);
664                             p2 = p;
```

```
665                     }
666                     else
667                             ajoutercoup(lscoup, coord - i*10);
668             }
669
670             for(i=1 ; (p.grille[o - i][a - i])%2 == -1 && coordvalide(coord - i*11) ; i++){
671                     if(mode){
672                             deplp(&p2, coord, coord - i*11);
673                             if(!echec(p2,d ? 0 : 1))
674                                     ajoutercoup(lscoup, coord - i*11);
675                             p2 = p;
676                     }
677                     else
678                             ajoutercoup(lscoup, coord - i*11);
679             }
680             if((p.grille[o - i][a - i])%2 == d && coordvalide(coord - i*11)){
681                     if(mode){
682                             deplp(&p2, coord, coord - i*11);
683                             if(!echec(p2,d ? 0 : 1))
684                                     ajoutercoup(lscoup, coord - i*11);
685                             p2 = p;
686                     }
687                     else
688                             ajoutercoup(lscoup, coord - i*11);
689             }
690             for(i=1 ; (p.grille[o - i][a + i])%2 == -1 && coordvalide(coord - i*9) ; i++){
691                     if(mode){
692                             deplp(&p2, coord, coord - i*9);
693                             if(!echec(p2,d ? 0 : 1))
694                                     ajoutercoup(lscoup, coord - i*9);
695                             p2 = p;
696                     }
697                     else
698                             ajoutercoup(lscoup, coord - i*9);
```

```
699                 }
700                 if((p.grille[o - i][a + i])%2 == d && coordvalide(coord - i*9)){
701                         if(mode){
702                                 deplp(&p2, coord, coord - i*9);
703                                 if(!echec(p2,d ? 0 : 1))
704                                         ajoutercoup(lscoup, coord - i*9);
705                                 p2 = p;
706                         }
707                         else
708                                 ajoutercoup(lscoup, coord - i*9);
709                 }
710
711                 for(i=1 ; (p.grille[o][a - i])%2 == -1 && coordvalide(coord - i) ; i++){
712                         if(mode){
713                                 deplp(&p2, coord, coord - i);
714                                 if(!echec(p2,d ? 0 : 1))
715                                         ajoutercoup(lscoup, coord - i);
716                                 p2 = p;
717                         }
718                         else
719                                 ajoutercoup(lscoup, coord - i);
720                 }
721                 if((p.grille[o][a - i])%2 == d && coordvalide(coord - i)){
722                         if(mode){
723                                 deplp(&p2, coord, coord - i);
724                                 if(!echec(p2,d ? 0 : 1))
725                                         ajoutercoup(lscoup, coord - i);
726                                 p2 = p;
727                         }
728                         else
729                                 ajoutercoup(lscoup, coord - i);
730                 }
731
732                 for(i=1 ; (p.grille[o][a + i])%2 == -1 && coordvalide(coord + i) ; i++){
```

```
733                     if(mode){
734                             deplp(&p2, coord, coord + i);
735                             if(!echec(p2,d ? 0 : 1))
736                                     ajoutercoup(lscoup, coord + i);
737                             p2 = p;
738                     }
739                     else
740                             ajoutercoup(lscoup, coord + i);
741             }
742             if((p.grille[o][a + i])%2 == d && coordvalide(coord + i)){
743                     if(mode){
744                             deplp(&p2, coord, coord + i);
745                             if(!echec(p2,d ? 0 : 1))
746                                     ajoutercoup(lscoup, coord + i);
747                             p2 = p;
748                     }
749                     else
750                             ajoutercoup(lscoup, coord + i);
751             }
752
753             for(i=1 ; (p.grille[o + i][a - i])%2 == -1 && coordvalide(coord + i*9) ; i++){
754                     if(mode){
755                             deplp(&p2, coord, coord + i*9);
756                             if(!echec(p2,d ? 0 : 1))
757                                     ajoutercoup(lscoup, coord + i*9);
758                             p2 = p;
759                     }
760                     else
761                             ajoutercoup(lscoup, coord + i*9);
762             }
763             if((p.grille[o + i][a - i])%2 == d && coordvalide(coord + i*9)){
764                     if(mode){
765                             deplp(&p2, coord, coord + i*9);
766                             if(!echec(p2,d ? 0 : 1))
```

```
767                             ajoutercoup(lscoup, coord + i*9);
768                     p2 = p;
769             }
770             else
771                     ajoutercoup(lscoup, coord + i*9);
772     }
773
774     for(i=1 ; (p.grille[o + i][a + i])%2 == -1 && coordvalide(coord + i*11) ; i++){
775             if(mode){
776                     deplp(&p2, coord, coord + i*11);
777                     if(!echec(p2,d ? 0 : 1))
778                             ajoutercoup(lscoup, coord + i*11);
779                     p2 = p;
780             }
781             else
782                     ajoutercoup(lscoup, coord + i*11);
783     }
784     if((p.grille[o + i][a + i])%2 == d && coordvalide(coord + i*11)){
785             if(mode){
786                     deplp(&p2, coord, coord + i*11);
787                     if(!echec(p2,d ? 0 : 1))
788                             ajoutercoup(lscoup, coord + i*11);
789                     p2 = p;
790             }
791             else
792                     ajoutercoup(lscoup, coord + i*11);
793     }
794
795     for(i=1 ; (p.grille[o + i][a])%2 == -1 && coordvalide(coord + i*10) ; i++){
796             if(mode){
797                     deplp(&p2, coord, coord + i*10);
798                     if(!echec(p2,d ? 0 : 1))
799                             ajoutercoup(lscoup, coord + i*10);
800                     p2 = p;
```

```
801                     }
802                     else
803                             ajoutercoup(lscoup, coord + i*10);
804             }
805             if((p.grille[o + i][a])%2 == d && coordvalide(coord + i*10)){
806                     if(mode){
807                             deplp(&p2, coord, coord + i*10);
808                             if(!echec(p2,d ? 0 : 1))
809                                     ajoutercoup(lscoup, coord + i*10);
810                             p2 = p;
811                     }
812                     else
813                             ajoutercoup(lscoup, coord + i*10);
814             }
815             break;
816
817             case 4: //Tour Blanche
818             d = 1;
819
820             case 5: //Tour Noire
821             for(i=1 ; (p.grille[o - i][a])%2 == -1 && coordvalide(coord - i*10) ; i++){
822                     if(mode){
823                             deplp(&p2, coord, coord - i*10);
824                             if(!echec(p2,d ? 0 : 1))
825                                     ajoutercoup(lscoup, coord - i*10);
826                             p2 = p;
827                     }
828                     else
829                             ajoutercoup(lscoup, coord - i*10);
830             }
831
832             if((p.grille[o - i][a])%2 == d && coordvalide(coord - i*10)){
833                     if(mode){
834                             deplp(&p2, coord, coord - i*10);
```

```
835                     if(!echec(p2,d ? 0 : 1))
836                             ajoutercoup(lscoup, coord - i*10);
837                     p2 = p;
838             }
839             else
840                     ajoutercoup(lscoup, coord - i*10);
841     }
842
843     for(i=1 ; (p.grille[o][a - i])%2 == -1 && coordvalide(coord - i) ; i++){
844             if(mode){
845                     deplp(&p2, coord, coord - i);
846                     if(!echec(p2,d ? 0 : 1))
847                             ajoutercoup(lscoup, coord - i);
848                     p2 = p;
849             }
850             else
851                     ajoutercoup(lscoup, coord - i);
852     }
853     if((p.grille[o][a - i])%2 == d && coordvalide(coord - i)){
854             if(mode){
855                     deplp(&p2, coord, coord - i);
856                     if(!echec(p2,d ? 0 : 1))
857                             ajoutercoup(lscoup, coord - i);
858                     p2 = p;
859             }
860             else
861                     ajoutercoup(lscoup, coord - i);
862     }
863
864     for(i=1 ; (p.grille[o][a + i])%2 == -1 && coordvalide(coord + i) ; i++){
865             if(mode){
866                     deplp(&p2, coord, coord + i);
867                     if(!echec(p2,d ? 0 : 1))
868                             ajoutercoup(lscoup, coord + i);
```

```
869                         p2 = p;
870                 }
871                 else
872                         ajoutercoup(lscoup, coord + i);
873         }
874         if((p.grille[o][a + i])%2 == d && coordvalide(coord + i)){
875                 if(mode){
876                         deplp(&p2, coord, coord + i);
877                         if(!echec(p2,d ? 0 : 1))
878                                 ajoutercoup(lscoup, coord + i);
879                         p2 = p;
880                 }
881                 else
882                         ajoutercoup(lscoup, coord + i);
883         }
884
885         for(i=1 ; (p.grille[o + i][a])%2 == -1 && coordvalide(coord + i*10) ; i++){
886                 if(mode){
887                         deplp(&p2, coord, coord + i*10);
888                         if(!echec(p2,d ? 0 : 1))
889                                 ajoutercoup(lscoup, coord + i*10);
890                         p2 = p;
891                 }
892                 else
893                         ajoutercoup(lscoup, coord + i*10);
894         }
895         if((p.grille[o + i][a])%2 == d && coordvalide(coord + i*10)){
896                 if(mode){
897                         deplp(&p2, coord, coord + i*10);
898                         if(!echec(p2,d ? 0 : 1))
899                                 ajoutercoup(lscoup, coord + i*10);
900                         p2 = p;
901                 }
902                 else
```

```
903                     ajoutercoup(lscoup, coord + i*10);
904             }
905             break;
906
907             case 6: //Fou Blanc
908             d = 1;
909
910             case 7: //Fou Noir
911             for(i=1 ; (p.grille[o - i][a - i])%2 == -1 && coordvalide(coord - i*11) ; i++){
912                     if(mode){
913                             deplp(&p2, coord, coord - i*11);
914                             if(!echec(p2,d ? 0 : 1))
915                                     ajoutercoup(lscoup, coord - i*11);
916                             p2 = p;
917                     }
918                     else
919                             ajoutercoup(lscoup, coord - i*11);
920             }
921             if((p.grille[o - i][a - i])%2 == d && coordvalide(coord - i*11)){
922                     if(mode){
923                             deplp(&p2, coord, coord - i*11);
924                             if(!echec(p2,d ? 0 : 1))
925                                     ajoutercoup(lscoup, coord - i*11);
926                             p2 = p;
927                     }
928                     else
929                             ajoutercoup(lscoup, coord - i*11);
930             }
931
932             for(i=1 ; (p.grille[o - i][a + i])%2 == -1 && coordvalide(coord - i*9) ; i++){
933                     if(mode){
934                             deplp(&p2, coord, coord - i*9);
935                             if(!echec(p2,d ? 0 : 1))
936                                     ajoutercoup(lscoup, coord - i*9);
```

```
937                         p2 = p;
938                 }
939                 else
940                         ajoutercoup(lscoup, coord - i*9);
941         }
942         if((p.grille[o - i][a + i])%2 == d && coordvalide(coord - i*9)){
943                 if(mode){
944                         deplp(&p2, coord, coord - i*9);
945                         if(!echec(p2,d ? 0 : 1))
946                                 ajoutercoup(lscoup, coord - i*9);
947                         p2 = p;
948                 }
949                 else
950                         ajoutercoup(lscoup, coord - i*9);
951         }
952
953         for(i=1 ; (p.grille[o + i][a - i])%2 == -1 && coordvalide(coord + i*9) ; i++){
954                 if(mode){
955                         deplp(&p2, coord, coord + i*9);
956                         if(!echec(p2,d ? 0 : 1))
957                                 ajoutercoup(lscoup, coord + i*9);
958                         p2 = p;
959                 }
960                 else
961                         ajoutercoup(lscoup, coord + i*9);
962         }
963         if((p.grille[o + i][a - i])%2 == d && coordvalide(coord + i*9)){
964                 if(mode){
965                         deplp(&p2, coord, coord + i*9);
966                         if(!echec(p2,d ? 0 : 1))
967                                 ajoutercoup(lscoup, coord + i*9);
968                         p2 = p;
969                 }
970                 else
```

```
971                     ajoutercoup(lscoup, coord + i*9);
972             }
973
974             for(i=1 ; (p.grille[o + i][a + i])%2 == -1 && coordvalide(coord + i*11) ; i++){
975                     if(mode){
976                             deplp(&p2, coord, coord + i*11);
977                             if(!echec(p2,d ? 0 : 1))
978                                     ajoutercoup(lscoup, coord + i*11);
979                             p2 = p;
980                     }
981                     else
982                             ajoutercoup(lscoup, coord + i*11);
983
984             }
985             if((p.grille[o + i][a + i])%2 == d && coordvalide(coord + i*11)){
986                     if(mode){
987                             deplp(&p2, coord, coord + i*11);
988                             if(!echec(p2,d ? 0 : 1))
989                                     ajoutercoup(lscoup, coord + i*11);
990                             p2 = p;
991                     }
992                     else
993                             ajoutercoup(lscoup, coord + i*11);
994             }
995             break;
996
997             case 8: //Cavalier Blanc
998             c = 0;
999
1000            case 9: //Cavalier Noir
1001            if((p.grille[o - 2][a - 1])%2 != c && coordvalide(coord - 21)){
1002                    if(mode){
1003                            deplp(&p2, coord, coord - 21);
1004                            if(!echec(p2,c))
```

```
1005                                    ajoutercoup(lscoup, coord - 21);
1006                            p2 = p;
1007                    }
1008                    else
1009                            ajoutercoup(lscoup, coord - 21);
1010            }
1011            if((p.grille[o - 2][a + 1])%2 != c && coordvalide(coord - 19)){
1012                    if(mode){
1013                            deplp(&p2, coord, coord - 19);
1014                            if(!echec(p2,c))
1015                                    ajoutercoup(lscoup, coord - 19);
1016                            p2 = p;
1017                    }
1018                    else
1019                            ajoutercoup(lscoup, coord - 19);
1020            }
1021            if((p.grille[o - 1][a - 2])%2 != c && coordvalide(coord - 12)){
1022                    if(mode){
1023                            deplp(&p2, coord, coord - 12);
1024                            if(!echec(p2,c))
1025                                    ajoutercoup(lscoup, coord - 12);
1026                            p2 = p;
1027                    }
1028                    else
1029                            ajoutercoup(lscoup, coord - 12);
1030            }
1031            if((p.grille[o - 1][a + 1])%2 != c && coordvalide(coord - 8)){
1032                    if(mode){
1033                            deplp(&p2, coord, coord - 8);
1034                            if(!echec(p2,c))
1035                                    ajoutercoup(lscoup, coord - 8);
1036                            p2 = p;
1037                    }
1038                    else
```

```
1039                    ajoutercoup(lscoup, coord - 8);
1040            }
1041            if((p.grille[o + 1][a - 2])%2 != c && coordvalide(coord + 8)){
1042                    if(mode){
1043                            deplp(&p2, coord, coord + 8);
1044                            if(!echec(p2,c))
1045                                    ajoutercoup(lscoup, coord + 8);
1046                            p2 = p;
1047                    }
1048                    else
1049                            ajoutercoup(lscoup, coord + 8);
1050            }
1051            if((p.grille[o + 1][a + 2])%2 != c && coordvalide(coord + 12)){
1052                    if(mode){
1053                            deplp(&p2, coord, coord + 12);
1054                            if(!echec(p2,c))
1055                                    ajoutercoup(lscoup, coord + 12);
1056                            p2 = p;
1057                    }
1058                    else
1059                            ajoutercoup(lscoup, coord + 12);
1060            }
1061            if((p.grille[o + 2][a - 1])%2 != c && coordvalide(coord + 19)){
1062                    if(mode){
1063                            deplp(&p2, coord, coord + 19);
1064                            if(!echec(p2,c))
1065                                    ajoutercoup(lscoup, coord + 19);
1066                            p2 = p;
1067                    }
1068                    else
1069                            ajoutercoup(lscoup, coord + 19);
1070            }
1071            if((p.grille[o + 2][a + 1])%2 != c && coordvalide(coord + 21)){
1072                    if(mode){
```

```
1073                    deplp(&p2, coord, coord + 21);
1074                    if(!echec(p2,c))
1075                            ajoutercoup(lscoup, coord + 21);
1076                    p2 = p;
1077            }
1078            else
1079                    ajoutercoup(lscoup, coord + 21);
1080    }
1081    break;
1082
1083    case 10: //Pion Blanc
1084    if(coord/10 == 6){
1085            if(p.grille[o - 2][a] == -1 && p.grille[o - 1][a] == -1){
1086                    if(mode){
1087                            deplp(&p2, coord, coord - 20);
1088                            if(!echec(p2,0))
1089                                    ajoutercoup(lscoup, coord - 20);
1090                            p2 = p;
1091                    }
1092                    else
1093                            ajoutercoup(lscoup, coord - 20);
1094            }
1095    }
1096    if(p.grille[o - 1][a] == -1){
1097            if(mode){
1098                    deplp(&p2, coord, coord - 10);
1099                    if(!echec(p2,0))
1100                            ajoutercoup(lscoup, coord - 10);
1101                    p2 = p;
1102            }
1103            else
1104                    ajoutercoup(lscoup, coord - 10);
1105    }
1106    if((p.grille[o - 1][a - 1])%2 == 1 && coordvalide(coord - 11)){
```

```
1107                    if(mode){
1108                            deplp(&p2, coord, coord - 11);
1109                            if(!echec(p2,0))
1110                                    ajoutercoup(lscoup, coord - 11);
1111                            p2 = p;
1112                    }
1113                    else
1114                            ajoutercoup(lscoup, coord - 11);
1115            }
1116            if((p.grille[o - 1][a + 1])%2 == 1 && coordvalide(coord - 9)){
1117                    if(mode){
1118                            deplp(&p2, coord, coord - 9);
1119                            if(!echec(p2,0))
1120                                    ajoutercoup(lscoup, coord - 9);
1121                            p2 = p;
1122                    }
1123                    else
1124                            ajoutercoup(lscoup, coord - 9);
1125            }
1126            break;
1127
1128            case 11: //Pion Noir
1129            if(coord/10 == 1){
1130                    if(p.grille[o + 2][a] == -1 && p.grille[o + 1][a] == -1){
1131                            if(mode){
1132                                    deplp(&p2, coord, coord + 20);
1133                                    if(!echec(p2,1))
1134                                            ajoutercoup(lscoup, coord + 20);
1135                                    p2 = p;
1136                            }
1137                            else
1138                                    ajoutercoup(lscoup, coord + 20);
1139                    }
1140            }
```

```
1141                    if(p.grille[o + 1][a] == -1){
1142                            if(mode){
1143                                    deplp(&p2, coord, coord + 10);
1144                                    if(!echec(p2,1))
1145                                            ajoutercoup(lscoup, coord + 10);
1146                                    p2 = p;
1147                            }
1148                            else
1149                                    ajoutercoup(lscoup, coord + 10);
1150                    }
1151                    if((p.grille[o + 1][a - 1])%2 == 0 && coordvalide(coord + 9)){
1152                            if(mode){
1153                                    deplp(&p2, coord, coord + 9);
1154                                    if(!echec(p2,1))
1155                                            ajoutercoup(lscoup, coord + 9);
1156                                    p2 = p;
1157                            }
1158                            else
1159                                    ajoutercoup(lscoup, coord + 9);
1160                    }
1161                    if((p.grille[o + 1][a + 1])%2 == 0 && coordvalide(coord + 11)){
1162                            if(mode){
1163                                    deplp(&p2, coord, coord + 11);
1164                                    if(!echec(p2,1))
1165                                            ajoutercoup(lscoup, coord + 11);
1166                                    p2 = p;
1167                            }
1168                            else
1169                                    ajoutercoup(lscoup, coord + 11);
1170                    }
1171                    break;
1172
1173                    default:
1174                    break;
```

```
1175            }
1176    }
1177
1178    void initlscoup(int lscoup[27]){
1179            int i;
1180            for(i=0 ; i<27 ; i++)
1181                    lscoup[i] = -1;
1182    }
1183
1184    void ajoutercoup(int lscoup[27], int coord){
1185            int i;
1186            for(i=0 ; lscoup[i] != -1 && i<27 ; i++){}
1187            lscoup[i] = coord;
1188    }
1189
1190    void afflscoup(int lscoup[27]){
1191            int i;
1192            for(i=0 ; lscoup[i] != -1 && i<27 ; i++){
1193                    printf("%c%d␣",itoc(lscoup[i]%10),((lscoup[i]/10)-8)*-1);
1194            }
1195            printf("\n");
1196    }
1197
1198    char itoc(int n){
1199            switch(n){
1200                    case 0:
1201                    return 'A';
1202                    case 1:
1203                    return 'B';
1204                    case 2:
1205                    return 'C';
1206                    case 3:
1207                    return 'D';
1208                    case 4:
```

```
1209                    return 'E';
1210                    case 5:
1211                    return 'F';
1212                    case 6:
1213                    return 'G';
1214                    default:
1215                    return 'H';
1216            }
1217    }
1218
1219    int echec(plateau p, int couleur){ //c (0 = blanc / 1 = noir)
1220            int r = -1;
1221            int i,j;
1222            int lscoup[27];
1223            for(i=0 ; i<8 && r == -1 ; i++){
1224                    for(j=0 ; j<8 && r == -1 ; j++){
1225                            if(p.grille[i][j] == couleur){
1226                                    r = i*10 + j;
1227                            }
1228                    }
1229            }
1230            int t = p.grille[r/10][r%10];
1231            if (t == 0){ //Roi blanc
1232                    if(     (p.grille[r/10 - 1][r%10]     == 1    && coordvalide(r - 10)) ||
1233                            (p.grille[r/10 - 1][r%10 - 1] == 1    && coordvalide(r - 11)) ||
1234                            (p.grille[r/10 - 1][r%10 + 1] == 1    && coordvalide(r - 9))  ||
1235                            (p.grille[r/10][r%10 - 1]     == 1    && coordvalide(r - 1))  ||
1236                            (p.grille[r/10][r%10 + 1]     == 1    && coordvalide(r + 1))  ||
1237                            (p.grille[r/10 + 1][r%10 - 1] == 1    && coordvalide(r + 9))  ||
1238                            (p.grille[r/10 + 1][r%10 + 1] == 1    && coordvalide(r + 11)) ||
1239                            (p.grille[r/10 + 1][r%10]     == 1    && coordvalide(r + 10))) {
1240                            return 1;
1241                    }
1242            }
```

```
1243        if (t == 1){ //Roi blanc
1244                if(     (p.grille[r/10 - 1][r%10]     == 0    && coordvalide(r - 10)) ||
1245                        (p.grille[r/10 - 1][r%10 - 1] == 0    && coordvalide(r - 11)) ||
1246                        (p.grille[r/10 - 1][r%10 + 1] == 0    && coordvalide(r - 9))  ||
1247                        (p.grille[r/10][r%10 - 1]     == 0    && coordvalide(r - 1))  ||
1248                        (p.grille[r/10][r%10 + 1]     == 0    && coordvalide(r + 1))  ||
1249                        (p.grille[r/10 + 1][r%10 - 1] == 0    && coordvalide(r + 9))  ||
1250                        (p.grille[r/10 + 1][r%10 + 1] == 0    && coordvalide(r + 11)) ||
1251                        (p.grille[r/10 + 1][r%10]     == 0    && coordvalide(r + 10))) {
1252                        return 1;
1253                }
1254        }
1255
1256        if(couleur)
1257                couleur = 0;
1258        else couleur = 1;
1259
1260        for(i=0 ; i<8 ; i++){
1261                for(j=0 ; j<8 ; j++){
1262                        if((p.grille[i][j])%2 == couleur && p.grille[i][j] != couleur){
1263                                initlscoup(lscoup);
1264                                coupslegaux(p, i*10 + j, lscoup, 0);
1265                                for(int x = 0 ; lscoup[x] != -1 ; x++){
1266                                        if (lscoup[x] == r){
1267                                                return 1;
1268                                        }
1269                                }
1270                        }
1271                }
1272        }
1273        return 0;
1274 }
1275
1276 int mat(plateau p, int couleur){ //c (0 = blanc / 1 = noir)
```

```
1277        int i,j,k;
1278        int lscoup[27];
1279        plateau p2;
1280        p2 = p;
1281        for(i=0 ; i<8 ; i++){
1282                for(j=0 ; j<8 ; j++){
1283                        if(p.grille[i][j]%2 == couleur){
1284                                initlscoup(lscoup);
1285                                coupslegaux(p2, i*10 + j, lscoup, 0);
1286                                for(k=0 ; lscoup[k] != -1 ; k++){
1287                                        deplp(&p2, i*10 + j, lscoup[k]);
1288                                        if(!echec(p2, couleur)){
1289                                                return 0;
1290                                        }
1291                                        p2 = p;
1292                                }
1293                        }
1294                }
1295        }
1296        return 1;
1297 }
1298
1299 arbre initarbre(){
1300        arbre a = malloc(sizeof(noeud));
1301        a->coup = -1;
1302        a->suiv[0] = NULL;
1303        return a;
1304 }
1305
1306 void inserefeuille (arbre a, int coord, int c){
1307        int i;
1308        arbre b;
1309        for (i = 0;; i++){
1310                if (! (a->suiv[i])) {
```

```
1311                            b = malloc (sizeof(noeud));
1312                            b->coord = coord;
1313                            b->coup = c;
1314                            b->suiv[0] = NULL;
1315                            a->suiv[i] = b;
1316                            a->suiv[i+1] = NULL;
1317                            return;
1318                    }
1319            }
1320    }
1321
1322    void inserearbre (arbre a, arbre b, int coord, int c){
1323            int i;
1324            for (i = 0;; i++){
1325                    if (! (a->suiv[i])){
1326                            a->suiv[i] = b;
1327                            a->suiv[i]->coord = coord;
1328                            a->suiv[i]->coup = c;
1329                            a->suiv[i+1]=NULL;
1330                            return;
1331                    }
1332            }
1333    }
1334
1335    void afficherarbre(arbre a, int nbc){
1336            int i;
1337            for(i=0 ; a->suiv[i]!=NULL ; i++){
1338                    if(i)
1339                            printf("OU\n");
1340                    printf("coup␣%d␣:␣%c%d␣va␣en␣%c%d\n", nbc, itoc(a->suiv[i]->coord%10), 8 - a->suiv[i
                           ]->coord/10, itoc(a->suiv[i]->coup%10), 8 - a->suiv[i]->coup/10);
1341                    afficherarbre(a->suiv[i], nbc + 1);
1342            }
1343            if(!a->suiv[0])
```

```
1344                printf("Echec et mat !!!\n");
1345 }
```