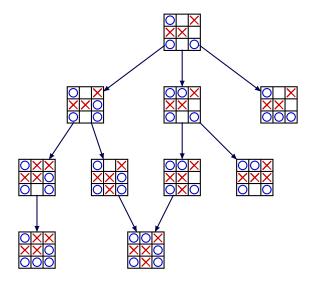
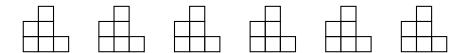
Jeux

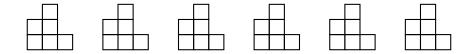
Exercice 1.3:



Exercice 1.4









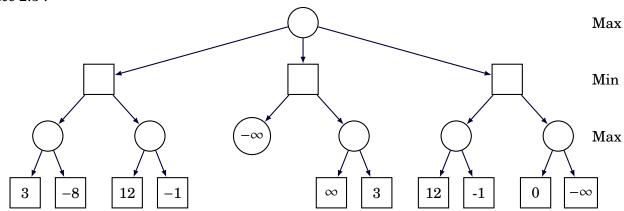
Exercice 2.1:

```
1
   let rec min_max jr etat
2
        match coups jr etat with
3
        | [] -> let g = gagnant etat in
4
                if g = 1 then -1 else g+1
5
        | l -> let f, deb = if jr = 0 then max, ref (-1) else min, ref 1 in
6
                let h successeur =
7
8
9
                    deb := f !deb v
10
11
                in
12
                List.iter h l;
13
                !deb
```

Exercice 2.2:

```
let heuristique etat =
 1
 2
        {\tt match} etat.(1).(1) with
 3
        | '0' -> 50
 4
        | 'X' -> -50
5
        | _ -> 0
6
 7
    let rec min_max jr etat nbcoups =
 8
        match .
9
                   -> let g = gagnant etat in
        ١.
10
                     if g = 1 then -100 else 100*(g+1)
11
12
                   -> let f, deb = if jr = 0 then max, ref (-100) else min, ref 100 in
13
                     let h successeur =
                         let v = min_max (1-jr) successeur (nbcoups-1) in
14
15
                         deb := f !deb v
16
                     in
17
                     List.iter h l;
18
                     !deb
```

Exercice 2.3:



Élagage alpha-beta:

```
1
    let rec min_max jr etat nbcoups alpha beta =
 2
        match
                                       with
 3
                  -> let g = gagnant etat in
                if g = 1 then -100 else 100*(g+1)
 4
 5
 6
                  -> let deb = if jr = 0 then ref (-100) else ref 100 in
 7
               let h successeur =
                     if jr = 0 then begin
 8
9
                         if not (!deb >= beta) then begin
10
                             let v = min_max (1-jr) successeur (nbcoups-1) (max !deb alpha) beta in
11
                             deb := max v !deb
12
                         end end;
13
                     if jr = 1 then begin
14
                         if not (!deb <= alpha) then begin</pre>
15
                             let v = min_max (1-jr) successeur (nbcoups-1) alpha (min !deb beta) in
16
                             deb := min v !deb
17
                         end end
18
                in
19
                List.iter h l;
20
                !deb
21
22 let rec alphabeta jr etat nbcoups = min_{\rm max} jr etat nbcoups (-100) 100
```