Si le cœur vous en dit, vous pouvez tout faire ou refaire. Plus raisonnablement, il vous est demandé de programmer les fonctions manipulant le tableau de jeu `Glob.gb`, à savoir:

- `val init_gb : unit -> unit`
  Initialise aléatoirement le tableau de jeu `Glob.gb`.
- `val end_game : unit -> bool`
  Vaut `true` si la partie est terminée: il ne reste plus aucune sélection possible.
- `val mark_sel : int * int -> unit`
  Marque, dans `Glob.gb`, la sélection déterminée par la pièce en `(icol, ilig)`.
- `val collapse_down : int array -> unit`
  Supprime les pièces marquées de la colonne passée en argument et compacte les pièces restantes.
- `val collapse : unit -> unit`
  Supprime les pièces marquées du tableau de jeu `Glob.gb` et compacte les colonnes restantes.

Pour ce, vous complèterez les d'efinitions absentes du fichier `jeu.ml` qui vous est fourni.

Afin de faciliter la mise au point de votre travail, nous vous fournissons également quelques fonctions pour l'affichage en mode texte dans le fichier `tli.ml`.

Losrque tout sera au point, vous pourrez compiler l'ensemble de l'application avec la commande:

```
ocamlc -custom -o jeu graphics.cma glob.cmo gui.cmo jeu.ml
```

qui produira un exécutable nommé `jeu`.