Algorithmique et probabilités : le jeu du lièvre et de la tortue

Règle du jeu. À chaque tour, on lance un dé. Si le 6 sort, alors le lièvre gagne la partie, sinon la tortue avance d'une case. La tortue gagne quand elle a avancé 6 fois.
Question : le jeu est-il à l'avantage du lièvre ou de la tortue ?

a. Algorithme 1 : Simulation d'une partie sans boucle

La partie se finit en au plus six lancés. Il est donc possible de simuler une partie sans avoir recours à une boucle.

Variables
    dé : la face du dé tirée au hasard
    tour : compte le nombre de tours que dure la partie
Initialisation
    dé prend une valeur entière aléatoire entre 1 et 6 compris
    tour prend la valeur 1
Traitement
    Si dé < 6 alors
    | dé prend une valeur entière aléatoire entre 1 et 6 compris
    | tour augmente de 1
    └
    Si dé < 6 alors
    | dé prend une valeur entière aléatoire entre 1 et 6 compris
    | tour augmente de 1
    └
    Si dé < 6 alors
    | dé prend une valeur entière aléatoire entre 1 et 6 compris
    | tour augmente de 1
    └
    Si dé < 6 alors
    | dé prend une valeur entière aléatoire entre 1 et 6 compris
    | tour augmente de 1
    └
    Si dé < 6 alors
    | dé prend une valeur entière aléatoire entre 1 et 6 compris
    | tour augmente de 1
    └
Sortie
    Si dé = 6 alors
    | Affiche « Le lièvre gagne »
    sinon
    | Affiche « La tortue gagne »
    └
    Affiche tour

1) Réaliser cette algorithme sur le tableur Open Office.
Ingrédients :
`ALEA.ENTRE.BORNES`(*nombre1*;*nombre2*) → génère un nombre entier aléatoire compris entre *nombre1* et *nombre2*.
`NB.SI`(*cellule1*:*cellule2*;*nombre1*) → compte le nombre d'apparition de *nombre1* entre *cellule1* et *cellule2*.
`SI`(*condition*;*valeur_si_vrai*;*valeur_si_faux*) → si condition est vérifiée, affiche *valeur_si_vrai*, sinon, affiche *valeur_si_faux*.

2) Réaliser cet algorithme avec Algobox.

Ingrédients :
`Ajouter SI... ALORS` → pour créer une boucle « si » (avec un « sinon » éventuellement)
`random()` → Obtenir un nombre pseudo-aléatoire compris en 0 et 1 :
`floor(x)` → Partie entière d'une variable x
`Déclarer nouvelle variable ; AFFECTER valeur à variable ; AFFICHER message ; AFFICHER variable`…

b. Algorithme 2 : Cumuler un grand nombre d'expériences.

Il suffit d'aménager le programme afin d'insérer la simulation d'une partie dans une boucle et de compter le nombre de parties gagnées par le lièvre ou la tortue.

Variables
- dé : la face du dé tirée au hasard
- N : le nombre de parties à simuler
- k : le compteur de boucle
- tortue : le nombre de parties gagnées par la tortue

Initialisation
- tortue prend une valeur 0

Traitement
- Pour k de 1 à N
  - dé prend une valeur entière aléatoire entre 1 et 6 compris
  - Si dé < 6 alors
    - dé prend une valeur entière aléatoire entre 1 et 6 compris
  - Si dé < 6 alors
    - dé prend une valeur entière aléatoire entre 1 et 6 compris
  - Si dé < 6 alors
    - dé prend une valeur entière aléatoire entre 1 et 6 compris
  - Si dé < 6 alors
    - dé prend une valeur entière aléatoire entre 1 et 6 compris
  - Si dé < 6 alors
    - dé prend une valeur entière aléatoire entre 1 et 6 compris
  - Si dé < 6 alors
    - tortue prend la valeur tortue + 1

Sortie

Affiche tortue.

Variante améliorée : faire afficher « la tortue gagne … fois sur ... » avec les bonnes valeurs.

Réaliser cet algorithme avec Algobox
Ingrédient :
`POUR... DE... A...` → pour créer une boucle « Pour ».

c. Algorithme 3 : Avec une structure itérative conditionnelle.

Évidemment, plutôt que de répéter 6 fois les mêmes instructions, il est possible de simuler une partie à l'aide d'une boucle.
De cette façon, il sera facile d'expérimenter de nouveaux jeux en modifiant le nombre de cases que doit parcourir la tortue.

```
Variables
    dé : la face du dé tirée au hasard
    case : le numéro de la case sur laquelle se trouve la tortue
    N : le nombre de cases que doit parcourir la tortue pour gagner.
Initialisation
    N prend la valeur 6
    case prend la valeur 0.
Traitement
    Répète
    |    dé prend une valeur entière aléatoire entre 1 et 6 inclus.
    |    Si dé < 6 alors
    |    |    case prend la valeur case + 1
    |    |
    |
    └jusqu'à [ dé = 6 ou case = N ]
Sortie
    Si case = N alors
    |    Affiche « La tortue gagne »
    Sinon
    |    Affiche « Le lièvre gagne »
```

Réaliser cet algorithme avec Algobox.
Ingrédient :
TANT QUE... → Pour créer une boucle « Répète ...»