La loi géométrique tronquée sur un exemple

I) Présentation du jeu

Pour jouer vous devez miser 10 €. Vous lancez un dé.
Si vous faites 6 au premier lancer,vous êtes remboursés de votre mise et vous gagnez en plus 5 €. Le jeu s'arrête.
Si vous faites 6 au deuxième lancer,vous êtes remboursés de votre mise et vous gagnez en plus 4 €. Le jeu s'arrête.
Si vous faites 6 au troisième lancer,vous êtes remboursés de votre mise et vous gagnez en plus 3 €. Le jeu s'arrête.
Si vous faites 6 au quatrième lancer,vous êtes remboursés de votre mise et vous gagnez en plus 2 €. Le jeu s'arrête.
Si vous faites 6 au cinquième lancer,vous êtes remboursés de votre mise et vous gagnez en plus 1 €. Le jeu s'arrête.
Si vous faites 6 au sixième lancer,vous êtes juste remboursés de votre mise. Le jeu s'arrête.
Si vous n'avez toujours pas fait de 6 après 6 lancers, le jeu s'arrête et vous avez perdu votre mise.

II) Les questions à se poser

a) Faire l'inventaire des gains algébriques possibles à ce jeu.

b) Trouver une relation simple liant le rang du lancer où le premier six est apparu et le gain algébrique du joueur.

c) A quelles conditions continue-t-on à lancer le dé ?

d) A quelles conditions le jeu s'arrête t-il ?

III) Construction d'un premier algorithme

On veut simuler une partie et obtenir l'affichage du rang d'apparition du premier 6 ainsi que le gain algébrique du joueur.
On aura besoin de déclarer trois variables :
rang_du_lancer : on y mettra le rang du lancer du dé
dé : on y mettra un nombre aléatoire entre 0 et 1. On conviendra que « faire un 6 » se traduira par : random() > $\frac{5}{6}$.
gain : on y mettra le gain algébrique du joueur.
Compléter l'algorithme suivant :

initialisation :
rang_du_lancer **prend la valeur** 0
dé **prend la valeur** 0

traitement :
Tant que ☐ **et** ☐
rang_du_lancer **prend la valeur** ☐
dé **prend la valeur** random()

Si *dé* > $\frac{5}{6}$ **Alors**
gain **prend la valeur** ☐
Sinon
rang_du_lancer **prend la valeur** 7 *(arbitraire)*
gain **prend la valeur** ☐

sortie :
Afficher *rang_du_lancer*
Afficher *gain*

IV) Construction d'un deuxième algorithme

Nous allons maintenant simuler 20 000 parties et calculer le gain moyen par partie.afin de deviner si le jeu est favorable ou défavorable au joueur.

Pour *numéro_partie* **Allant de** 1 **à** 20000
  rang_du_lancer **prend la valeur** 0
  dé **prend la valeur** 0

  Tant que *dé* < 5/6 **et** *rang_du_lancer* < 6
    rang_du_lancer **prend la valeur** *rang_du_lancer* + 1
    dé **prend la valeur** random()

  Si *dé* > $\frac{5}{6}$ **Alors**
    gain **prend la valeur** 6 - *rang_du_lancer*
  Sinon
    gain **prend la valeur** -10

  gain_total **prend la valeur** ☐

gain_moyen **prend la valeur** ☐

Afficher *gain_moyen*

V) Ecriture du deuxième algorithme à partir du premier

- Déclarer les trois nouvelles variables : *numéro_partie*, *gain_total*, *gain_moyen*.
- Construire la boucle correspondant au 20 000 parties.
- Réinitialiser les valeurs de *rang_du_lancer* et de *dé* à 0

Il est possible de déplacer les différents blocs qui suivent (correspondant à une partie) dans la boucle en travaillant avec **Edition, copier, coller**

- Faire les déplacements (deux déplacements, un pour la boucle conditionnelle, un autre pour le test).
- Terminer l'écriture : Attention, bien réfléchir à l'endroit où l'on place l'affectation de *gain_total* et celle de *gain_moyen*.