1 Un jeu équitable

On considère le jeu de hasard entre un joueur A et un joueur B :

- Pour chaque partie, les deux joueurs lancent chacun un dé à six faces de manière simultanée :
  - ➢ si la somme des faces est paire, le joueur A gagne la partie, ce qui lui rapporte 1 point ;
  - ➢ si la somme des faces est impaire, le joueur B gagne la partie, ce qui lui rapporte 1 point ;
- Le premier qui a **8 points** est le vainqueur du jeu.

On a construit une fonction de simulation d'une partie pour ce jeu en langage Python :

Code Python :

```
import random

def partie():
    """simulation du lancer de chaque dé par A et B, avec victoire pour A si la
    somme des faces est paire, victoire pour B si la somme est impaire"""
    lancer_A = random.randint(1,6)
    lancer_B = random.randint(1,6)
    somme = lancer_A + lancer_B
    if somme % 2 == 0:
        return "A"
    else:
        return "B"
```

On a vérifié par simulation que chaque partie de ce jeu était **équitable** et que chaque joueur avait une probabilité de $\frac{1}{2}$ de gagner la partie.

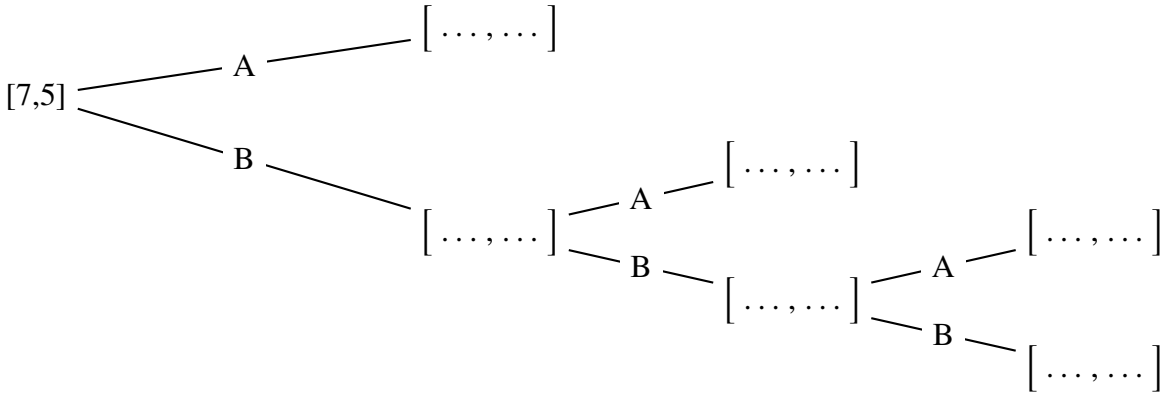
2 Rappel du problème du jeu interrompu

Deux joueurs A et B jouent au jeu précédent après avoir misé chacun 42 € : la mise en jeu est donc de 84 €.
Le premier qui a **8 points** est le vainqueur du jeu et il gagne la totalité de la **mise** de 84 €.
Seulement, A et B sont obligés de s'arrêter avant d'avoir pu terminer le jeu : quand ils s'arrêtent, A a **7 points** et B a **5 points**. Avant de se séparer, ils veulent se partager la mise puisque personne ne l'a complètement gagnée.

Combien doit se faire le partage pour tenir compte de l'état du jeu au moment de l'interruption ?

2.1 Fins de jeux

1. Compléter l'arbre ci-dessous pour recenser les fins de jeux possibles :



2. Compléter alors la fonction `deroule_jeu` qui simule la fin du jeu :

Code Python :

```
import random
def deroule_jeu():
    """simule la fin du jeu lorsqu'il manque une victoire à A et 3
    victoires à B"""
    lancer = ..................... # nouvelle partie
    if ..................... :
        return ..... # A a gagné la partie qu'il lui manquait et il a gagné
        le jeu [8,5]
    else: # B a gagné la partie et il lui manque 2 victoires et 1 pour A
    [7,6]
        lancer = .....................
        if ..................... :# A a gagné la partie qu'il lui manquait et
        il a gagné le jeu [8,6]
            return .....
        else: # B a gagné la partie et il lui manque 1 victoire et 1 pour A
        [7,7]
            lancer = .....................
            if .....................:# A a gagné la partie qu'il lui manquait
            et il a gagné le jeu [8,7]
                return .....
            else:# B a gagné la partie et il a gagné le jeu [7,8]
                return .....
```

3. Compléter la fonction `simulation_jeux` qui va simuler un grand nombre de fois le déroulé du jeu et renvoyer la fréquence de victoires de A et de B :

Code Python :

```
def simulation_jeux(nb_jeux):
    """Simulation d'un grand nombre de jeux pour connaitre la fréquence de
    victoires de chaque joueur"""
    victoires_A = 0 # compteur de victoires de A
    victoires_B = 0 # compteur de victoires de B
    for simu in range(nb_jeux): # boucle pour répéter la simulation
        jeu = ..................... # appel de la fonction deroule_jeu
        if ..................... : # si le jeu mène à une victoire de A
            victoires_A = ......................... # le compteur de
            victoires de A augmente de 1
        else: # si le jeu mène à une victoire de B
            victoires_B = ......................... # le compteur de
            victoires de B augmente de 1
    return (............................ , ............................) #
    renvoie la fréquence de victoires pour chaque joueur
```