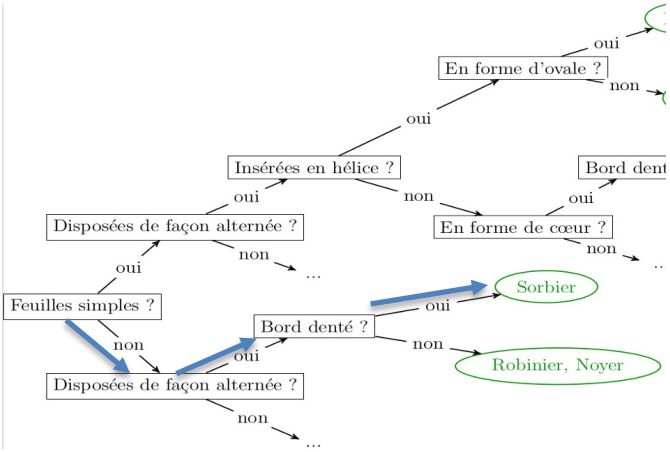
Amérique du Nord – 2025 – sujet1 - Correction

Exercice 1 (6 points)

---

1. Identification à partir de l'arbre de decision

Folia simples, alternées, insérées en hélice et non ovales : le seul végétal correspondant dans l'arbre est le **Sorbier**.



---

2. Construction de l'arbre de décision arbre_2

Folia complexes, alternées et à bord denté : aucun végétal connu ne correspond exactement à ces caractéristiques dans l'arbre. Le végétal ne peut donc pas être identifié.

---

3. Construction de l'arbre de décision arbre_2

```
feuille1 = Feuille_resultat(['Sorbier'])
feuille2 = Feuille_resultat(['Robinier', 'Noyer'])
feuille_vide = Feuille_resultat([])
noeud1= Noeud('Bord denté ?', feuille1, feuille2)
noeud2 = Noeud('Alternées ?', noeud1, feuille_vide)
arbre_2 = Noeud('Simples ?', feuille_vide, noeud2)
```

👉*Commentaire : on construit l'arbre de bas en haut en respectant exactement la structure de l'arbre de décision fourni dans l'énoncé.*

On pouvait aussi imbriquer les nœuds :

```
arbre_2 = Noeud(
   "Simples ?",
   Feuille_resultat(["Tilleul"]),
   Noeud(
      "Alternées ?",
      Noeud(
         "Bord denté ?",
         Feuille_resultat(["Sorbier"]),
         Feuille_resultat(["Robinier", "Noyer"])
        ),
      Feuille_resultat([]),
     )
  )
```

---

4. Méthode est_resultat `classe Noeud`

```
class Noeud:
    def est_resultat(self):
        return False
```

👉*Commentaire : un nœud correspond à une question, ce n'est pas un résultat.*

---

5. Méthode est_resultat classe `Feuille_resultat`

```
class Feuille_resultat:
    def est_resultat(self):
        return True
```

👉 *Commentaire : Une feuille correspond toujours à un résultat.*

---

6. Méthode nb_vegetaux `classe Feuille_resultat`

```
class Feuille_resultat:
    def nb_vegetaux(self):
        return len(self.vegetaux)
```

---

7. Méthode nb_vegetaux `classe Noeud`

```
class Noeud:
    def nb_vegetaux(self):
        return self.sioui.nb_vegetaux() + self.sinon.nb_vegetaux()
```

👉*Commentaire : on additionne récursivement le nombre de végétaux accessibles par chacune des branches.*

---

8. Méthode liste_questions `classe Feuille_resultat`

```
class Feuille_resultat:
    def liste_questions(self):
        return []
```

---

9. Méthode `liste_questions` `classe Noeud`

```
class Noeud:
    def liste_questions(self):
        return [self.question] + self.sioui.liste_questions() +
self.sinon.liste_questions()
```

👉*Commentaire : on ajoute la question courante puis toutes celles accessibles dans les sous-arbres.*

---

10. Fonction `est_bien_renseigne`

```
def est_bien_renseigne(dico_vegetal, arbre):
    for question in arbre.liste_questions():
        if question not in dico_vegetal:
            return False
    return True
```

👉Commentaire : on vérifie que chaque question présente dans l'arbre est bien une clé du dictionnaire.

---

11. Fonction `identifier_vegetaux`

```
def identifier_vegetaux(dico_vegetal, arbre):
    if arbre.est_resultat():
        return arbre.vegetaux
    if dico_vegetal[arbre.question]:
        return identifier_vegetaux(dico_vegetal, arbre.sioui)
    else:
        return identifier_vegetaux(dico_vegetal, arbre.sinon)
```

👉*Commentaire : on parcourt récursivement l'arbre en suivant les réponses contenues dans le dictionnaire.*
