P11 : Contrôle final (correction)

Exercice 1 : un chouette exercice non réaliste ! (6 points)

Question 1 (2 points)

Il y a deux manières d'interpréter le calcul : soit la reproduction a lieu avant de retirer les 6 chouettes, soit elle a lieu après.

Dans le **premier cas**, pour chaque année, on prend le nombre de chouettes divisé par 2 (pour obtenir le nombre de couples), on le multiplie par 8 (pour obtenir le nombre de naissances) et on ajoute le résultat au nombre de chouette existantes. Enfin, si l'année est divisible par 3, alors on retire 6 à la somme précédente.

On introduit 10 chouettes composées de 10/2 = 5 couples.
On a donc 5 * 8 œufs pondus pendant la 1er année.
Après 1 ans, il y a 10 + 40 = 50 chouettes.

On a 50 chouettes donc 50/2 = 25
Il y a 25 * 8 = 200 naissances au cours de l'année.
Après 2 ans, on a donc 50 + 200 = 250 chouettes.

On a 250 donc 250/2 = 125 couples chouettes au début de l'année 3.
Il y a 125 * 8 = 1000 naissances au cours de l'année.
L'année est multiple de 3 donc on retire 6 chouettes soit 1000 – 6 = 994.
Après 3 ans, il a y 250 + 994 = 1244 chouettes.

Il y a 1244 chouettes donc 1244/2 = 622 couples au début de l'année 4.
Il y a 622 * 8 = 4976 naissances au cours de l'année.
Après 4 ans, il y a 1244 + 4976 = 6220 chouettes.

Dans le **second cas**, pour chaque année, on vérifie d'abord si l'année est divisible par 3 et si c'est le cas on retire 6 au nombre de chouettes actuelles. Ensuite, on divise le nombre de chouettes par 2 et on multiplie par 8 et on ajoute ce résultat au nombre de chouettes.

On introduit 10 chouettes composées de 10/2 = 5 couples.
On a donc 5 * 8 œufs pondus pendant la 1er année.
Après 1 ans, il y a 10 + 40 = 50 chouettes.

On a 50 chouettes donc 50/2 = 25
Il y a 25 * 8 = 200 naissances au cours de l'année.
Après 2 ans, on a donc 50 + 200 = 250 chouettes.

On a 250 donc 250/2 = 125 couples chouettes au début de l'année 3.
L'année est multiple de 3 donc on retire 6 chouettes soit 250 – 6 = 244 avec 244/2 = 122 couples.
Il y a 122 * 8 = 976 naissances au cours de l'année.
Après 3 ans, il a y 244 + 976 = 1220 chouettes.

Il y a 1220 chouettes donc 1220/2 = 610 couples au début de l'année 4.
Il y a 610 * 8 = 4880 naissances au cours de l'année.
Après 4 ans, il y a 1220 + 4880 = 6100 chouettes.

Question 2 (3 points)

Fonction : Chouettes
Précondition : annee ≥ 0
Entrée : - annee : entier
Sortie : - nChouettes : entier
Début
| **Soit** nChouettes = 10
| **Pour** n **de** 1 **à** annee **avec un pas de** +1 **Faire**
| | **Si** n est divisible par 3 **Alors**
| | | **Soit** nChouettes = nChouettes – 6
| | **Fin Si**
| | **Soit** nChouettes = nChouettes + (nChouettes / 2) * 8
| **Fin Pour**
| **Retourner** nChouettes
Fin

Note : on peut aussi accepter **Divisible(n, 3)** ou **n % 3 == 0**

Question 3 (1 point)

Procédure : Main
Entrée : void
Local : /
Sortie : void
Début
| **Pour** annee **de** 1 **à** 10 **avec un pas de** +1 **Faire**
| | **Afficher**(**Chouettes**(annee))
| **Fin Pour**
Fin

Exercice 2 : Pokémon, attrapez-les tous ! (9 points)

Question 1 (1 point)

```
Soit structure Pokemon
{
        Champ nom : chaîne de caractères
        Champ niveau : entier
        Champ attaque : entier (ou réel)
        Champ defense : entier (ou réel)
        Champ pointsDeVie : entier
}
```

Question 2 (1 point)

```
Soit structure Pokemon
{
        Constructeur : Pokemon
        Entrée :        - xNom : chaîne de caractères
                        - xNiveau : entier
                        - xAttaque : entier (ou réel)
                        - xDefense : entier (ou réel)
                        - xPointsDeVie : entier
        Sortie :        void
        Début
        |       Soit nom = xNom
        |       Si xNiveau ≤ 0 Alors
        |       |       Soit niveau = 1
        |       Sinon
        |       |       Soit niveau = xNiveau
        |       Fin Si
        |       Soit attaque = xAttaque
        |       Soit defense = xDefense
        |       Si xPointsDeVie ≤ 0 Alors
        |       |       Soit pointsDeVie = 1
        |       Sinon
        |       |       Soit pointsDeVie = xPointsDeVie
        |       Fin Si
        Fin
}
```

Question 3 (2 points)

```
Soit jeu = Dictionnaire<chaîne de caractères, Liste<structure Pokemon>>
```

Question 4 (2 points)

```
Procédure : AfficherDictionnaire
Entrée :        - jeu : Dictionnaire<chaîne de caractères, Liste<structure Pokemon>>
Local :         /
Sortie :        void
Début
|       Pour Chaque chaîne de caractères nomDresseur dans jeu.Clefs() Faire
|       |       Afficher(“Deck du dresseur” + nomDresseur + “\n”)
|       |       Pour Chaque structure Pokemon p dans jeu[nomDresseur] Faire
|       |       |       Afficher(Pokémon + p.nom)
|       |       |       Afficher(“ niveau = ” + p.niveau)
|       |       |       Afficher(“ points de vie = ”+ p.pointsDeVie)
|       |       |       Afficher(“ attaque = ” + p.attaque)
|       |       |       Afficher(“ defense = ” + p.defense + “\n”)
|       |       Fin Pour Chaque
|       Fin Pour Chaque
Fin
```

Question 5 (2 points)

```
Fonction : CalculerNiveauMax
Précondition : nomDresseur appartient à la liste des clefs de jeu
Entrée :        - jeu : Dictionnaire<chaîne de caractères, Liste<structure Pokemon>>
                - nomDresseur : chaîne de caractères
Local :         /
Sortie :        - niveauMax : entier
Début
|       Soit niveauMax = -1
|       Si jeu.Clefs().Contient(nomDresseur) Alors
|       |       Pour Chaque structure Pokemon p dans jeu[nomDresseur] Faire
|       |       |       Si p.niveau > niveauMax Alors
|       |       |       |       Soit niveauMax = p.niveau
|       |       |       Fin Si
|       |       Fin Pour Chaque
|       Fin Si
|       Retourner niveauMax
Fin
```

Question 6 (1 point)

```
Procédure : AfficherNiveauxMax
Entrée :        - jeu : Dictionnaire<chaîne de caractères, Liste<structure Pokemon>>
Local :         - niveauMaxDresseur : entier
Sortie :        void
Début
|       Pour Chaque chaîne de caractères nomDresseur dans jeu.Clefs() Faire
|       |       Soit niveauMaxDresseur = CalculerNiveauMax(jeu, nomDresseur)
|       |       Afficher(nomDresseur + " : " + niveauDresseurMax + "\n")
|       Fin Pour Chaque
Fin
```

Exercice 3 : Tournoi de Pokémons (5 points + 2 points bonus)

Question 1 (5 points)

```
Fonction : SimulerCombatPokemons
Entrée :        - p1 : structure Pokemon
                - p2 : structure Pokemon
Local :         - pdv1, pdv2 : entiers
                - degats1, degats2 : entiers
                - action : entier
Sortie :        - vainqueur : structure Pokemon
Début
|       Soit action = 0 // décide qui doit attaquer (0 pour p1 et 1 pour p2)
|       Soit vainqueur = p1
|       Si p2.niveau < p1.niveau Alors
|       |       Soit action = 1
|       Fin Si
|       Soit pdv1 = p1.pointsDeVie
|       Soit pdv2 = p2.pointsDeVie
|       Soit degats1 = p1.niveau + Max(0, p1.attaque – p2.defense)
|       Soit degats2 = p2.niveau + Max(0, p2.attaque – p1.defense)
|       Tant Que pdv1 > 0 ET pdv2 > 0 Faire
|       |       // attaque de p1 sur p2
|       |       Si action == 0 Alors
|       |       |       Soit pdv2 = pdv2 – degats1
|       |       |       Soit action = 1
|       |       // attaque de p2 sur p1
|       |       Sinon Si action == 1 Alors
|       |       |       Soit pdv1 = pdv1 - degats2
|       |       |       Soit action = 0
|       |       Fin Si
|       Fin Tant Que
|       Si pdv1 ≤ 0 Alors
|       |       Soit vainqueur = p2
|       Fin Si
|       Retourner vainqueur
Fin
```

Note : on peut accepter que les champs p1.pointsDeVie et p2.pointsDeVie soient modifiés. L'énoncé ne l'interdit pas mais l'exemple donné pour illustrer ne les modifie pas.

Question bonus (2 points)

Fonction : SimulerCombatDresseurs
Précondition : nom1 et nom2 appartiennent à la liste des clefs de jeu
Entrée : - jeu : Dictionnaire<chaîne de caractères, Liste<structure Pokemon>>
- nom1 : chaîne de caractères
- nom2 : chaîne de caractères
- vaincus : Liste<**structure Pokemon**> modifiable
Local : - vaincus1 : Liste<**structure Pokemon**>
- vaincus2 : Liste<**structure Pokemon**>
Sortie : - vainqueur : chaîne de caractères

```
Début
|   Si jeu.Clefs().Contient(nom1) == FAUX OU jeu.Clefs().Contient(nom2) == FAUX Alors
|   |   Exception/Erreur…
|   Fin Si
|   Soit vaincus1 = Liste<structure Pokemon> vide
|   Soit vaincus2 = Liste<structure Pokemon> vide
|   Soit vainqueur = nom1
|   Tant Que jeu[nom1].Taille() > 0 ET jeu[nom2].Taille() > 0 Faire
|   |   Soit p1 = jeu[nom1][0]
|   |   Soit p2 = jeu[nom2][0]
|   |   Soit pVainqueur = SimulerCombatPokemons(p1, p2)
|   |   Si pVainqueur == p1 Alors
|   |   |   vaincus2.Ajouter(p2)
|   |   |   jeu[nom2].Supprimer(0)
|   |   Sinon Si pVainqueur == p2 Alors
|   |   |   vaincus1.Ajouter(p1)
|   |   |   jeu[nom1].Supprimer(0)
|   |   Fin Si
|   Fin Tant Que
|   Si jeu[nom2].Taille() == 0 Alors
|   |   Soit vainqueur = nom1
|   |   Pour Chaque structure Pokemon p dans vaincus2 Faire
|   |   |   vaincus.Ajouter(p)
|   |   Fin Pour Chaque
|   Sinon Si jeu[nom1].Taille() == 0 Alors
|   |   Soit vainqueur = nom2
|   |   Pour Chaque structure Pokemon p dans vaincus1 Faire
|   |   |   vaincus.Ajouter(p)
|   |   Fin Pour Chaque
|   Fin Si
|   Retourner vainqueur
Fin
```