420-JV4-AS Concepts et Programmation Objet 1 7 fevrier 2018
Felix Soumpholphakdy

Travaux Pratiques 3

Objectifs:

- Utiliser les instructions de sélection (if, else)
- Utiliser les instructions d'itération (for, while)
- Manipuler les fonctions et les classes
- Apprendre à utiliser une fonction récursive
- Apprendre a manipuler une liste chaînée
- Afficher un résultat dans la console

Critères de performances:

- 3.1 Résolution efficace des problèmes de conception de l'algorithme.
- 3.2 Résolution efficace des problèmes de traduction de l'algorithme dans le langage de programmation.
- 3.3 Résolution efficace des problèmes d'utilisation du langage de programmation.
- 4.3 Interprétation juste des résultats.
- 4.4 Fonctionnement correct du programme.

Date de remise: 21 février 2018

Instructions:

1. Téléchargez le projet Visual Studio ici: https://github.com/felixsoum/420JV4AS-TP3
2. Écrivez votre nom en commentaire au début du fichier Program.cs
3. Faites tout le travail dans Program.cs et soumettre seulement ce fichier sur Léa

Chaque question correspond à une fonction. Les fonctions doivent être implémenté correctement pour que l'appel de ces fonctions affichent des résultats qui correspondent à ceux dans ce document.

Question 1. Récursivité

Une fonction récursive est une fonction qui fait appel à elle-même. L'implémentation de certain algorithmes se facilite par la récursivité. Pour cette question, il ne faut pas utiliser les instructions d'itération.

(a) La suite de Fibonacci est une suite d'entiers où chaque terme est égale a la somme des deux termes précédents. Voici les premiers termes: 0, 1, 1, 2, 3, 5, 8, 13, (…). Implémentez une fonction récursive qui retourne le n-ième terme de la suite de Fibonacci.

Résultats:

```
// Console.WriteLine(Question1a(7));
13
```

```
// Console.WriteLine(Question1a(20));
6765
```

(b) Implémentez une fonction récursive qui inverse l'ordre des caractères d'une chaine.

Résultats:

```
// Console.WriteLine(Question1b("bonk"));
"knob"
```

```
// Console.WriteLine(Question1b("alucard"));
"dracula"
```

Question 2. Donjon

Un donjon peut se définir comme une collection de salles qui se connecte entre elles. Dans le projet, la classe Room est une classe de base qui possède 2 champs:

- name, le nom de la salle
- previous, la salle précédente
- next, la salle suivante

(a) Implémentez une fonction qui enlève la n-ième salle et reconnecte les salles correctement en ordre. Cette fonction à comme paramètre et comme valeur de retour la première salle. Une fonction qui imprime l'ordre des salles par leur nom est fournie.

Résultats:

```
// Print(Question2a(Room.CreateDungeon(), 2));
-[A]-[B]-[D]-[E]
```

```
// Print(Question2a(Room.CreateDungeon(), 0));
-[B]-[C]-[D]-[E]
```

(b) Implémentez une fonction qui inverse l'ordre des salles du donjon. La première salle devient la dernière salle.

Résultats:

```
// Print(Question2b(Room.CreateDungeon()));
-[E]-[D]-[C]-[B]-[A]
```

(c) Implémentez une fonction qui crée une copie de chaque salle et l'insère devant elle.

Résultats:

```
// Print(Question2c(Room.CreateDungeon()));
-[A]-[A]-[B]-[B]-[C]-[C]-[D]-[D]-[E]-[E]
```