Thème : Algorithmique	**Activité : Par ici la monnaie**	NSI 1ère

Lorsque vous passez à la caisse d'un magasin, il n'est pas rare que le caissier doive vous rendre de l'argent car le montant que vous lui avez donné est supérieur à celui que vous devez payer.

Supposons qu'on doive vous rendre la somme de 2,63€. Il y a bien évidemment énormément de possibilités pour vous rendre la dite somme. Il y a fort à parier que les solutions du type "263 pièces de 1 centime" ne vous conviennent pas, pour cause, personne n'a envie de remplir son porte-monnaie avec autant de pièces...

Le problème est donc de minimiser le nombre de pièces rendues pour un montant fixé.

Ici, on considère que l'ensemble des valeurs des pièces disponibles est le suivant (on considérera ici leur nombre illimité) :

{2€, 1€, 50¢, 20¢, 10¢, 5¢, 2¢, 1¢}

Partie I : Pour débuter ensemble

On considère deux cas de la situation décrite précédemment :

Cas 1 : Le caissier doit vous rendre la somme de 8 centimes.

Cas 2 : Le caissier doit vous rendre 2,63€

I.1. La première solution à laquelle on peut penser est d'énumérer toutes les combinaisons possibles, de sélectionner celles qui impliquent un minimum de pièces et de choisir la meilleure.
Pour le cas 1, énumérer toutes les combinaisons possibles et sélectionner les combinaisons impliquant un minimum de pièces.

I.2. La méthode naïve est-elle appropriée dans le cas 2 ? Pourquoi ?

I.3. Pour le cas 2, proposer une combinaison de pièces pour le rendu de monnaie impliquant le moins de pièces possible. Comment avez-vous fait pour trouver cette combinaison de manière efficace ? Décrire votre méthode étape par étape.

I.4. En déduire pourquoi cet algorithme est appelé « glouton » (qui signifie « prenant le maximum de ce qui est disponible ») ?

I.5. Généraliser votre démarche pour rendre n'importe quel montant de manière efficace. Pour cela, énumérer, étape par étape, la méthode à suivre (ce qui constituera l'algorithme). On notera la liste de l'ensemble des pièces disponibles (qu'on supposera prédéfini) *Pieces*, le montant à rendre *Montant* et la liste des pièces rendues *Monnaie.*

Partie II : A vos ordinateurs !

On considérera ici que :

- toutes les valeurs sont en centimes.
- l'ensemble des pièces disponibles est donné par : *Pieces = [1, 2, 5, 10, 20, 50, 100, 200]*

II.1. A l'aide de la méthode gloutonne, et sans utiliser l'ordinateur, donner la meilleure combinaison de pièces rendues (en centimes) pour les montants suivants :

Montant	Combinaison proposée
2,41€	1×200¢, 2×20¢, 1×1¢
0,97€	
5,63€	

II.2. Ecrire une fonction nommée *ValeurMaxListe* qui prend en arguments une liste *L* (qui n'est pas obligatoirement ordonnée) et un entier *N* et qui renvoie la valeur de l'élément de la liste ayant la plus grande valeur inférieure ou égale à *N*.

II.3. A partir de l'algorithme ci-dessous et en utilisant la fonction *ValeurMaxListe* écrite précédemment, écrire une fonction d'optimisation par méthode gloutonne, nommée *RenduMonnaie*, qui prend en arguments la liste *Pieces* des pièces disponibles et le montant *Montant* à rendre et qui renvoie la liste *Monnaie* des pièces rendues.

```
Entrées : Pièces, une liste d'entiers ; Montant, un nombre
Monnaie = une liste vide
TANT QUE Montant > 0
    Chercher dans la liste Pieces la plus grande valeur MaxVal inférieure ou égale à Montant
    Ajouter cette valeur MaxVal à la liste Monnaie
    Soustraire MaxVal de Montant
FIN TANT QUE
RENVOYER Monnaie
```

Tester votre code avec les exemples de la question II.1.

II.4. Pourquoi est-il nécessaire d'utiliser des nombres entiers plutôt que des flottants pour la liste *Pieces* des pièces disponibles ?

II.5. L'algorithme glouton est très facile à mettre en œuvre, mais présente une faiblesse.
Tester votre programme avec les entrées suivantes :

- *Pieces = [1, 3, 4]*
- *Montant = 6*

La solution trouvée est-elle optimale ? Quel est alors le point faible de l'algorithme glouton (le prix à payer pour sa simplicité) ?

Travail de réflexion à faire avant le prochain cours

Le problème du sac à dos des voleurs

Dans une maison se trouve des objets de valeur. Chaque objet *i* possède un prix de revente v_i et un poids w_i (tous deux des entiers naturels)
Des cambrioleurs viennent avec un sac à dos *S* et cherchent à prendre une partie de ces objets dans le sac à dos, qui dispose d'une capacité limitée *W*. Les voleurs cherchent à maximiser la somme des prix de revente des objets à mettre dans le sac à dos en respectant sa capacité maximale.
On considère ici qu'une fois l'objet mis dans le sac, il n'est plus disponible dans la liste des objets à prendre.
Le problème est donc :
Etant donnée une capacité W, on cherche un sous-ensemble d'objets qui maximise $\sum v_i$ *avec la contrainte* $\sum w_i \leq W$.

1) Combien de valeurs sont associées à chaque objet ? Quel type de variable est alors utilisé pour représenter l'ensemble des objets disponibles ?

2) Sous quelle condition considère-t-on que le sac est plein ?

3) Réfléchir à la démarche à suivre pour résoudre ce problème.

Ce qu'il faut retenir

❖ Algorithme glouton

La méthode gloutonne est utilisée pour déterminer la manière optimale de sélectionner des valeurs parmi une liste donnée en respectant certains critères de limitation.
Elle consiste à répéter, étape par étape, le choix de la plus grande valeur de la liste qui ne dépasse pas le critère de limitation.

Algorithme glouton – la base

```
Entrées : Limitation, un nombre ; Liste, une liste d'entiers
Selection = une liste vide
TANT QUE Limitation > 0
    Chercher dans la liste Liste la plus grande valeur inférieure ou égale à Limitation
    Ajouter cette valeur à la liste Selection
    Soustraire la valeur à celle de Limitation
FIN TANT QUE
RENVOYER Selection
```