Exercice 3 (8 points)

Cet exercice traite de tableaux, de dictionnaires, de récursivité et de programmation dynamique.

Une scierie possède un stock de planches. Chacune des planches du stock a une longueur entière (en mètre) comprise entre 1 m et 10 m et un prix entier (en euro) qui dépend de cette longueur. Le propriétaire de la scierie souhaite estimer la valeur marchande de son stock.

On utilise dans tout le problème la variable globale `prix` du type `list` telle que pour tout entier `n` compris entre 1 et 10, `prix[n]` est le prix de vente en euro d'une planche de longueur `n` (en mètre).

On suppose ici que la liste `prix` est `prix = [0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30]`. On remarque que l'élément d'indice 0 de `prix` est fixé à 0 mais ne sera jamais utilisé car une planche ne peut mesurer 0 m.

Partie B

On réalise qu'on peut parfois augmenter la valeur d'une planche en la découpant en planches plus petites (toujours de longueurs entières) et en vendant séparément les morceaux obtenus.

On cherche donc à déterminer la valeur maximale que peut rapporter une planche à l'entreprise, éventuellement en la découpant.

On remarque que :

- on ne peut pas découper une planche de 1 m : sa valeur maximale est donc 1€ ;
- une planche 2 m peut être vendue entière pour 5 € ou en deux planches de 1 m pour $1 + 1 = 2$ € : il vaut donc mieux ne pas la découper ;
- une planche de 3 m peut être vendue entière pour 8 € ou en une planche de 2 m et une planche de 1 m pour $1 + 5 = 6$ € ou en trois planches de 1 m pour $1 + 1 + 1 = 3$ € : il vaut donc mieux ne pas la découper.

7. Déterminer, en justifiant, la valeur maximale que peut rapporter une planche de 4 m.

On note `valmax(n)` la valeur maximale d'une planche de longueur `n` mètres pour `n` compris entre 1 m et 10 m en optimisant sa découpe et on propose la formule récursive ci-dessous.

- Si $n \leq 3$, `valmax(n) = prix[n]`.
- Sinon, `valmax(n) = max(prix[n], prix[n-1] + valmax(1), prix[n-2] + valmax(2), ... , prix[1] + valmax(n-1))`.

8. Expliquer le cas d'arrêt de cette formule récursive.
9. Recopier et compléter les quatre lignes ci-dessous.

```
- valmax(1) = prix[...] = ...
- valmax(2) = ... = ...
- valmax(3) = ... = ...
- valmax(4) = max(prix[...], prix[...] + ... ,
  prix[...] + ... , prix[...] + ...) = ...
```

10. Expliquer pourquoi, si $n \geq 4$, on a bien : `valmax(n) = max(prix[n], prix[n-1] + valmax(1), prix[n-2] + valmax(2), ... , prix[1] + valmax(n-1))`

On souhaite disposer de la fonction `valmax` qui prend en paramètre un entier `n` et qui renvoie la valeur maximale obtenue pour une planche de longueur `n` selon les prix de la variable globale `prix`.

11. Recopier et compléter les lignes 2 à 7 du code de la fonction `valmax` ci-après.

```
1 def valmax(n):
2   if ... :
3     ...
4   else:
5     maximum = prix[n]
6     for longueur in range(1, ...):
7       maximum = max(maximum, ... + ...)
8     return maximum
```

12. Expliquer pourquoi l'appel de `valmax(7)` effectuera deux appels de `valmax(5)`.