Informatique tronc commun
Devoir n° 2 – Partie rédigée

2 décembre 2017

Durée : 60 minutes, documents interdits.

Vous écrirez les fonctions demandées dans le langage `Python` (version 3) et rendrez sur papier votre composition.

Vous rédigerez soigneusement la ou les fonctions `Python` que vous devrez écrire, sans oublier les indentations, chaînes de documentation et commentaires nécessaires.

On numérotera chaque ligne de chaque bloc de code `Python`.

Lorsque vous écrirez un invariant ou un variant, vous ferez systématiquement référence aux lignes du bloc de code étudié. Par exemple : « Un invariant pour la boucle `for` des lignes n° 42 à 1515 est [...] ».

Lorsque vous justifierez un invariant ou un variant, vous ferez systématiquement référence à une ligne du bloc de code étudié. Par exemple : « au début de la ligne n° 42, on a [...] », ou « à la fin de la ligne n° 1515, on sait que [...] ».

Lorsque vous écrivez une fonction `Python`, on ne vous demande pas de vérifier que les arguments donnés sont corrects. Cependant, il sera apprécié d'indiquer les préconditions vérifiées par ces arguments dans la chaîne de documentation de la fonction.

Les fonctions `Python` permettant de faire un calcul sur les listes (ex : `max`, `sum`, *etc.*) ne sont pas autorisées, hormis la fonction `len`.

I. Questions de cours.

Q1 Que fait l'opérateur `and` en `Python` ? Sur quels types de valeurs l'opérateur s'applique-t-il et quelles sont ses propriétés remarquables ?

Q2 Quelles différences notables y a-t-il entre les types `tuple` et `list` en `Python` ?

II. Série harmonique.

Pour tout $n \in \mathbb{N}^*$, on définit

$$H_n = \sum_{k=1}^{n} \frac{1}{k}.$$

Q3 Écrire une fonction `H_depasse(M)` prenant en entrée un nombre `M` et renvoyant en sortie le plus petit entier naturel non nul n vérifiant $H_n \geq$ `M`.

III. Suite arithmético-géométrique.

On considère la fonction suivante.

```
1 def suite(n):
2     """Précondition : n entier naturel"""
3     u = 2
4     for i in range(n) :
5         u = 4 * u - 3
6     return u
```

Q4 Montrer que « $u = 4^i + 1$ » est un invariant d'entrée de boucle, pour la boucle for de la fonction `suite(n)`. En déduire le résultat renvoyé par cette fonction.

IV. Fonction mystère.

On considère la fonction suivante.

```
1 def mystere(n):
2     """Préconditions : n entier positif"""
3     L = []
4     c = 0
5     while c ** 2 <= n :
```

```
6           L.append(c**2)
7           c = c+1
8       return L
```

Q5 Montrer que, si `n` est un entier, un appel de la fonction `mystere(n)` renvoie un résultat, à l'aide d'un variant.

Q6 Que renvoie un appel de la fonction `mystere(n)` ? On justifiera la réponse, à l'aide notamment d'un invariant.

V. Pyramides dans un tableau.

Soit `t` $= [t_0, \dots, t_{n-1}]$ un tableau de nombres de longueur $n \in \mathbb{N}^*$, soit $k \in \mathbb{N}^*$ et $i \in [\![0, n+2-2k[\![$.

On dit que `t` possède une *pyramide* de *hauteur* k en position i si

$$t_i < t_{i+1} < \dots t_{i+k-1}$$

et si

$$t_{i+k-1} > t_{i+k} > \dots t_{i+2k-2}.$$

Par exemple, il y a une pyramide de longueur 1 en tout élément de `t`, et il y a une pyramide de longueur 2 en position i si

$$t_i < t_{i+1} \quad \text{et} \quad t_{i+1} > t_{i+2}.$$

Ainsi, avec

$$\texttt{t} = [-1, 0, 4, 2, -3, 0, 5, 1],$$

`t` a une pyramide de hauteur 3 en position 0, des pyramides de hauteur 2 en position 1 et 5 et des pyramides de hauteur 1 en toute position.

Q7 Écrire une fonction `nb_pyramides_2(t)` renvoyant le nombre de pyramides de hauteur 2 présentes dans le tableau de nombres `t` passé en argument.

Q8 Écrire une fonction `pyramide_a_partir(t,i)` prenant en argument un tableau de nombres `t` et un indice `i` et renvoyant la taille de la pyramide du tableau `t` débutant en position `i`.

Q9 Écrire une fonction `plus_haute_pyramide(t)` renvoyant la taille de la plus haute pyramide présente dans le tableau de nombres `t` passé en argument. On pourra utiliser la fonction `pyramide_a_partir(t,i)` de la question précédente, même si cette question n'a pas été résolue.