EXERCICE 1 (6 points)

Cet exercice porte sur l'exécution d'un programme Python et sur la décidabilité.

Dans cet exercice, on dira qu'un appel `f(x)`, où `f` est une fonction Python prenant un argument et `x` est une valeur, **termine** lorsque l'évaluation de `f(x)` renvoie toujours une valeur au bout d'un nombre fini d'étapes. A l'opposé, un tel appel ne termine pas s'il est possible qu'il effectue des instructions à l'infini.

Partie A : boucle `while`

Commençons par un premier exemple, avec une fonction prenant un entier en argument et utilisant une boucle `while`.

```
1 def f1(n):
2     i = n
3     while i != 10:
4         i = i + 1
5     return i
```

1. Sur votre copie, donner les valeurs successives de la variable `i` lors de l'exécution de `f1(7)`, et indiquer si `f1(7)` termine.

2. Indiquer si l'appel `f1(-2)` se termine. Si oui, indiquer la valeur renvoyée.

3. Sur votre copie, donner les 5 premières valeurs prises par la variable `i` lors de l'exécution de `f1(12)`, et indiquer si l'appel `f1(12)` va terminer ou non.

4. Préciser pour quels entiers `n` l'appel `f1(n)` se termine.

Partie B : fonction récursive

Prenons maintenant un deuxième exemple, avec une fonction récursive (prenant elle aussi un entier en argument).

```
1 def f2(n):
2     if n == 0:
3         return 0
4     else:
5         return n + f2(n-2)
```

5. L'appel `f2(4)` termine-t-il ? Si oui, indiquer la valeur renvoyée par `f2(4)` ; sinon, justifier brièvement.

6. L'appel `f2(5)` termine-t-il ? Si oui, indiquer la valeur renvoyée par `f2(5)` ; sinon, justifier brièvement.

7. Déterminer l'ensemble des entiers naturels `n` pour lesquels l'appel `f2(n)` termine.

8. Écrire une fonction Python `infini`, récursive, telle que l'appel `infini(n)` ne termine pour aucun entier `n`.

Partie C : le problème de l'arrêt

On se demande maintenant s'il est possible d'écrire une fonction `arret` qui prend en arguments une chaîne de caractères `code_f` contenant le code d'une fonction `f` et un argument `x` de `f`, et tel que `arret(code_f, x)` renvoie `True` si l'appel `f(x)` va terminer et `False` sinon.

Dans la suite de cet exercice, on suppose disposer d'une telle fonction `arret` et on implémente la fonction suivante, utilisant cette fonction `arret`, ainsi que la fonction `infini` de la question précédente dont l'appel `infini(n)` ne termine jamais quelle que soit la valeur de `n`.

```
1 def paradoxe(x):
2     if arret(x, x):
3         infini(42)
4     else:
5         return 0
```

De même, on suppose disposer d'une variable `code_paradoxe` contenant le code de la fonction `paradoxe` sous la forme d'une chaîne de caractères, et on s'intéresse à l'appel `paradoxe(code_paradoxe)`.

Cet appel de fonction commence par effectuer le test `arret(code_paradoxe, code_paradoxe)` dans le `if` de la ligne 2.

9. Dans le cas où `arret(code_paradoxe, code_paradoxe)` renvoie `True`, préciser la prochaine instruction à être exécutée. Dans ce cas, expliquer si l'appel `paradoxe(code_paradoxe)` termine.

10. Dans le cas où `arret(code_paradoxe, code_paradoxe)` renvoie `False`, préciser la prochaine instruction à être exécutée. Dans ce cas, expliquer si l'appel `paradoxe(code_paradoxe)` termine.

11. En déduire qu'une telle fonction `arret` ne peut exister.