**Activité 2 : Le flipper**

**Objectif de la séance :** Ecrire un algorithme de recherche d’une occurrence sur des valeurs de type quelconque.

**Situation 1 :**

Un groupe de 10 amis se retrouvent un samedi soir à *l’arcade game*. Ils souhaitent chacun leur tour jouer au flipper, chaque joueur possède un numéro allant de 1 à 10. L’ordre de passage est tiré au sort. Lorsqu’un joueur a fini sa partie, il inscrit son score dans une grille.

Le gérant de *l’arcade game* se rapproche du groupe et souhaite récompenser celui qui aura atteint le premier un score d’exactement 40 points. Par contre, le gérant est exténué, sa semaine a été longue. Il n’est plus capable de réfléchir correctement et souhaite trouver un moyen d’obtenir efficacement, et le plus rapidement possible, le numéro du joueur qui a atteint le score fixé.

**Travail demandé :**

*Grâce au matériel qui est à votre disposition (voir la liste ci-dessous), aidez le gérant de l’arcade game à trouver le joueur correspondant au score fixé. Ecrivez pour cela un algorithme en langage de programmation Python qui donne le numéro du joueur dont le score correspondant à ce qui est recherché.*

Document 1 : Grille des scores obtenus

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N° du joueur | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Score | 6 | 41 | 4 | 48 | 21 | 56 | 73 | 40 | 58 | 40 |

Document 2 : Logiciels EduPython,

Pyzo, Syder ou Jupyter notebook.

****

**Appelez le professeur pour valider.**

**Situation 2 :**

Le gérant est entièrement satisfait de votre travail. Il souhaite maintenant pouvoir modifier le score à sa guise et exécuter le programme. Ainsi il pourra, quelque soit le score qu’il aura choisi, obtenir le numéro du joueur à récompenser.

**Travail demandé :**

*Modifiez votre programme afin de satisfaire à nouveau le gérant de l’arcade game. Vous vous appuierez sur les mêmes documents de la situation 1.*

****

**Appelez le professeur pour valider.**

**Expertise des algorithmes :**

1. Evaluer la complexité des algorithmes écrits précédemment.
2. Démontrer la terminaison de l’algorithme de la situation 1.
3. Prouver la correction partielle de cet algorithme. Que peut-on alors conclure par rapport à la correction totale de cet algorithme ?

**Correction**

1. Une fois les algorithmes faits, on peut compter le nombre d’opérations (affectations, opérations arithmétiques, tests, itérations d’une boucle, affichage, renvois d’une fonction, etc) pour montrer que la complexité asymptotique des algorithmes utilisés est linéaire : *O(n).*

**Situation 1**

def occurence\_1(L):

n=len(L) *# (1 affectation)*

i=0 #on initialise i à zéro*(1 affectation)*

while ((i<n) and L[i]!=40): *# (n itérations + 2 tests / itération au pire des cas)*

i=i+1 #on incrémente i tant qu’on ne trouve pas le joueur gagnant *(1 affectation + 1 addition)*

if (i<n): #dans ce cas on trouve l’indice du joueur gagnant *(1 test)*

print(i+1, "est le numéro du joueur gagnant") *# (1 affichage + 1 addition)*

else: #dans ce cas on a parcouru toute la liste L *(1 test)*

print ("Aucun joueur a fait un score de 40") *# (1 affichage)*

Total :

**Situation 2**

def occurence\_2(L,x):

n=len(L)

i=0

while ((i<n) and L[i]!=x):

i=i+1

if (i<n):

print(i+1,"est le numéro du joueur gagnant")

else:

print ("Aucun joueur a fait un score de",x)

L’algorithme étant pratiquement le même que précédemment, puisqu’on a remplacé 40 par x, alors la complexité asymptotique est la même : *O(n)*.

1. **Preuve de terminaison**

On peut se contenter de prouver la terminaison et la correction partielle de la fonction *occurrence\_1()* puisque la fonction *occurrence\_2()* est basée sur le même algorithme.

Pour prouver la terminaison de la boucle « while », il suffit que les conditions de la boucle ne soient plus vérifiées. Ici, on arrête la boucle s’il existe un ***i*** compris entre ***0*** et ***n-1*** tel que ***L[i]=40*** (la condition ***L[i]≠40*** n’est plus vérifiée) ou bien si ***i=n*** (c’est donc la condition ***i<n*** qui n’est plus vérifiée). Le variant est donc ***n-i*** qui varie de ***n*** à ***0*** et garanti que la boucle se termine.

1. **Correction partielle**

Précondition :et pour tout

Postcondition : Trouver ***i*** tel ***L[i]=40*** ou affiche un message pour dire que ***40*** n’est pas dans la liste ***L***.

Invariant : et 40 n’est pas parmi les L[0 : i-1]

Condition : et

L’Invariant () et le contraire de la condition ( ou ) nous amène effectivement à la postcondition. (A noter que correspond au cas où l’élément n’est pas dans la liste considérée).

On vient ainsi de montrer la correction partielle.

Conclusion : Ayant démontré la terminaison de l’algorithme ET la correction partielle de celui-ci, on a montré la correction totale.