Université de Carthage Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul Département Mathématique-Informatique	IPEIN المعهد التحضيري للدراسات الهندسية بنابل Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul	Année universitaire : 2023/2024 Filière : SM, SP et ST Niveau d'étude : 2ème année Semestre : 1 Nombre de pages : 2 Date : 02/11/2023 Durée : 1H30

Devoir Surveillé n°1 d'informatique

*** Corrigé ***

Exercice1 : (10 points)

```
1.  #IPEIN 2023/2024 - DS1
2.  #Exercice1.py
3.  #Question 1 : (2.0 pts)
4.  def SIC_it(n):
5.      s = 0
6.      for i in range(1,n+1):
7.          s += 1/i**2
8.      return s
9.
10. #Question 2 : (2.0 pts)
11. def SIC_rec(n):
12.     if n == 1: return 1
13.     else: return 1/n**2 + SIC_rec(n-1)
14.
15. #Question 3 : (2.0 pts)
16. #
17. #   1/16+1/9+1/4+1 = 1.42
18. #     <-------
19. #              |
20. #              |     1/9+1/4+1 = 1.36
21. #      SIC_rec(4) <-------
22. #           |              |
23. #           |              |     1/4+1=1.25
24. #            -----> SIC_rec(3) <-------
25. #                         |              |
26. #                         |              |          1
27. #                          ------> SIC_rec(2) <-------
28. #                                        |              |
29. #                                        |              |
30. #                                         -----> SIC_rec(1)
31. #
32. # Dans ce cas la pile d'appels récursifs doit être de taille 3 cases
    mémoires.
33.
34. #Question 4 : (2.0 pts soit 0.5 par question)
35. #a- Calcul du coût et complexité temporelle pour SIC_it(n):
36. # Coût = 4n+2 soit une complexité temporelle linéaire O(n)
37.
38. #b- Calcul du coût et complexité spatiale pour SIC_it(n):
39. # cette fonction utilise 3 variables n, i et s
40. # d'où Coût = 3 unités de mémoire
41. # soit une complexité spatiale constante O(1)
42.
```

```
43.   #c- Calcul du coût et complexité temporelle pour SIC_rec(n):
44.   # SIC_rec(n) implique (n-1) appels récursifs et dans chaque appel il y a
45.   # 1 division + 1 puissance + 1 addition = 3 opérations (si on néglige
46.   # la comparaison et return de chaque appel)
47.   # d'où Coût = 3n soit une complexité temporelle linéaire O(n)
48.
49.   #d- Calcul du coût et complexité spatiale pour SIC_rec(n):
50.   # SIC_rec(n) implique (n-1) appels récursifs et pour chaque appel on
51.   # sauvegarde l'argument "n" dans la pile des appels (ici on néglige
52.   # l'adresse de retour)...
53.   # d'où Coût = n-1 cases mémoires dans la pile d'appels
54.   # soit une complexité spatiale linéaire O(n)
55.
56.   #Question 5 : (2.0 pts)
57.   from math import pi
58.   def main():
59.       while 1:
60.           try:
61.               n = int(input("Donner n>0:"))
62.               if n>0: break
63.           except: continue
64.       r = SIC_rec(n)
65.       pi_cal = (r*6)**0.5
66.       erreur = 100*(pi-pi_cal)/pi
67.       print("pi={:5.3f} avec un taux d'erreur =
      {:5.2f}%".format(pi_cal,erreur))
68.
```

Exercice2 : (10 points)

```
1.  #IPEIN 2023/2024 - DS1
2.  #Exercice2.py
3.  #Question 1 : (1.5 pts)
4.  def somDiv(n):
5.      """Retourne la somme des diviseurs propres de "n"."""
6.      somme = 1
7.      for i in range(2, n//2+1):
8.          if n % i == 0:
9.              somme += i
10.     return somme
11.
12. #Question 2 : (1.5 pts)
13. def estParfait(n):
14.     """Teste si "n" est parfait."""
15.     return somDiv(n) == n
16.
17. #Question 3 : (1.5 pts)
18. def estPremier(n):
19.     """Teste si "n" est premier."""
20.     return True if somDiv(n) == 1 else False
21.
22. #Question 4 : (1.5 pts)
23. def estChanceux(n):
24.     """Teste si "n" est chanceux."""
25.     if n<2: return False
26.     for i in range(2, n-1):
27.         if not estPremier(n + i + i**2):
28.             return False
29.     return True
30.
31. #Question 5 : (1.5 pts)
32. def SaisieRec(m):
33.     """Fonction de saisie d'un entier n"""
34.     try:
35.         n = int(input("Donner entier 0<n<="+str(m)+": "))
36.         if 0<n<=m: return n
37.         print("Erreur de saisie...")
38.         return SaisieRec(m)
39.     except:
40.         print("Erreur de conversion...")
41.         return SaisieRec(m)
42.
43. #Question 6 : (2.5 pts)
44. def main():
45.     n = SaisieRec(1000)
46.     try: f = open("Resultat.txt","w")
47.     except:
48.         print("Erreur fichier...")
49.         return
50.     LPremier = list()
51.     LParfait = list()
52.     LChanceux = []
53.     for i in range(1,n+1):
54.         if estPremier(i): LPremier.append(i)
55.         if estParfait(i): LParfait.append(i)
56.         if estChanceux(i): LChanceux.append(i)
57.     f.write("LPremier = " + str(LPremier) + "\n")
58.     f.write("LParfait = " + str(LParfait) + "\n")
59.     f.write("LChanceux = "+ str(LChanceux) + "\n")
60.     f.close()
61.
```