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

Exercice1 : (10 points)

Dans cet exercice on cherche à calculer une approximation de la valeur de π.

On rappelle que : $$\lim_{n\to+\infty} S_n = \lim_{n\to+\infty} \sum_{i=1}^{n} \frac{1}{i^2} = \frac{\pi^2}{6}.$$

D'où pour tout entier naturel « n » non nul $S_n = \sum_{i=1}^{n} \frac{1}{i^2}$. est une approximation de $\frac{\pi^2}{6}$

1- Ecrire « **`SIC_it(n)`** » une fonction Python **itérative** qui calcule la ***S**omme des **I**nverses des **C**arrés* des « n » premiers entiers naturels non nuls.
2- Ecrire « **`SIC_rec(n)`** » une fonction Python **récursive** qui calcule la même **S**omme des **I**nverses des **C**arrés des « n » premiers entiers naturels non nuls.
NB : Le cas de base « **condition d'arrêt** » est celui où la somme ne comporte qu'un terme (1 en l'occurrence) : $S_1 = 1$. C'est donc le cas où il n'y aura pas d'appel récursif.
3- Donnez le schéma d'exécution de la fonction récursive « **`SIC_rec(n)`** » pour n = 4. Déduire la taille de la pile nécessaire dans ce cas d'exécution.
4- Calcul de coût et complexité :
   a. Calculer le **coût temporel** de la fonction itérative « **`SIC_it(n)`** » et déduire sa **complexité**.
   b. Calculer le **coût spatial** de la fonction itérative « **`SIC_it(n)`** » et déduire sa **complexité**.
   c. Calculer le **coût temporel** de la fonction récursive « **`SIC_rec(n)`** » et déduire sa **complexité**.
   d. Calculer le **coût spatial** de la fonction récursive « **`SIC_rec(n)`** » et déduire sa **complexité**.
5- Donnez le code en python d'un programme principal « **`main()`** » qui permet :
   - Le saisie, avec tous les contrôles nécessaires, d'un entier « n » strictement positif,
   - De calculer la valeur de π par un appel de l'une de deux fonctions précédentes.
   - D'afficher la valeur de « **pi** » calculée et son **taux d'erreur** par rapport à « pi » de la bibliothèque « **math** » :
     NB : **taux d'erreur = 100*(pi – pi_calculée)/pi** , avec « pi » est celle qui se trouve dans la bibliothèque math.

```
>>> main()
Donner n>0:10
pi=3.049 avec un taux d'erreur =  2.94%
>>>
```

Exercice2 : (10 points)

– On appelle **nombre premier** tout entier naturel supérieur à 1 qui possède exactement deux diviseurs, l'unité et lui-même ;

– On appelle **diviseur propre** de « n », un diviseur quelconque de « n », avec « n » exclu ;

– Un entier naturel est dit **parfait** s'il est égal à la somme de tous ses diviseurs propres ;

– Les nombres « n » tels que : $(n + i + i^2)$ est premier pour tout « i » tel que $0 \leq i \leq (n - 2)$, sont appelés **nombres chanceux d'Euler** (Par exemple 2 est le premier nombre chanceux d'Euler).

1- Donner le code python de la fonction « **SomDiv(n)** » qui retourne la somme des diviseurs propre de son argument « n ».

2- Donner le code python de la fonction « **EstParfait(n)** » qui retourne un booléen « **True** » si « n » est un nombre parfait, « **False** » sinon.

3- Donner le code python de la fonction « **EstPremier(n)** » qui retourne un booléen « **True** » si « n » est un nombre premier, « **False** » sinon.

4- Donner le code python de la fonction « **EstChanceux(n)** » qui retourne un booléen « **True** » si « n » est un nombre chanceux d'Euler, « **False** » sinon.

5- Donner le code python d'une fonction **récursive** « **SaisieRec(m)** » qui saisit un entier « n » strictement positif et inférieur ou égal à « m ». C'est une fonction de saisie avec tous les contrôles nécessaires dont voici un exemple d'exécution :

```
>>> n = SaisieRec(10) ; n
Donner un entier 0<n<=10: 0
Erreur de saisie...
Donner un entier 0<n<=10: 5.4
Erreur de conversion...
Donner un entier 0<n<=10: 6
6
>>>
```

6- Ecrire un programme principal « **main()** » qui :

- Saisit un entier « n » avec $0<n\leq1000$,
- Ouvre un fichier « Resultat.txt » en mode écriture,
- Enregistre dans ce fichier les listes suivantes :
    - LPremiers : Liste des nombres premiers entre 1 et n,
    - LParfait : Liste des nombres parfaits entre 1 et n,
    - LChanceux : Liste des nombres chanceux entre 1 et n.

Bon courage…