ROYAUME DU MAROC

Ministère de l'Enseignement supérieur,
de la formation des cadres
et de la recherche scientifique

CONCOURS NATIONAL COMMUN
d'Admission aux
Grandes Écoles d'Ingénieurs
Marocaines et Assimilées

Sujet Blanc 2014

PREUVE D'INFORMATIQUE

Durée 2 heures

FILIÈRES : MP/ PSI/ TSI

Cette épreuve comporte 8 pages au format A4, en plus de cette page de garde.
L'usage de la calculatrice est interdit

L'énoncé de cette épreuve, commune aux candidats des filièresMP/ PSI/ TSI, comporte 8 pages.
L'usage de la calculatrice est interdit.

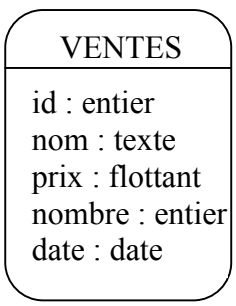
Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- *L'épreuve se compose de deux exercices et d'un problème.*
- *Toutes les instructions et les fonctions demandées seront écrites en langage Python.*
- *Les questions non traitées peuvent être admises pour aborder les questions ultérieures.*
- *Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions*

Exercice 1. Thème : base de données

Nous allons nous intéresser à la gestion d'une très petite entreprise qui vend du matériel informatique sur Internet comme des DVD, clefs USB, disques durs, etc.
Le responsable du magasin entrele bilan des composants vendus au cours de chaque journée dans une base de données nommée « *BD_ventes* ». Cette base de données est constituée d'une seule table nommée « ventes ». La description de cette table est la suivante :



Cette base de données sera une base de données de type*SQLite*.

Voici, à titre purement indicatif, la structure de cette table en *SQLite* :
("id" NOT NULL,"nom" TEXT,"prix" REAL,"nombre" INTEGER,"date" DATETIME)
dont voici un aperçu :

id	***nom***	***prix***	***nombre***	***date***
…	*…*	*…*	*…*	*…*
35	*DVD (par 10)*	*100*	*9*	*23/08/2014*
12	*clé USB 2go*	*20*	*4*	*25/08/2014*
13	*clé USB 8go*	*40*	*12*	*25/08/2014*
44	*DVD (par 100)*	*800*	*6*	*25/08/2014*
14	*clé USB 16go*	*50*	*50*	*26/08/2014*
14	*clé USB 16go*	*50*	*50*	*27/08/2014*
14	*clé USB 16go*	*50*	*50*	*28/08/2014*
35	*DVD (par 10)*	*100*	*9*	*23/08/2014*
…	*…*	*…*	*…*	*…*

La base de données (le fichier « *BD_ventes.sqlite* ») se trouve au même niveau que les pages constituant ce site web (sous C:\www). Dans cet exercice le contenu de cette base de données est déjà créé.

Question 1 :
Écrireune requête SQL donnant comme résultat la liste de tous les composants vendus un jour spécifique par exemple le 25/08/2014. Attention on ne souhaite pas afficher les doublons.

Question 2 :
Écrire une requête SQL donnant comme résultat la liste de tous les composants vendus sur une période, par exemple du 23/08/2014 inclus au 26/08/2014 inclus.
Vous pouvez utiliser une ou plusieurs requêtes *SQL*. Pour les doublons on fera le même traitement que lors de la première question.

Nous allons maintenant évaluer le bénéfice qu'a fait l'entreprise au cours du mois d'août sachant que pour chaque produit vendu la marge de bénéfice est de 30%.
Vous commencerez par définir la requête *SQL*, puis vous écrirez un programme en Python permettant de connaitre ce bénéfice pour le mois d'août.

Question 3 :
Expliciter la requête *SQL*.

Question 4 :
Écrive la fonction qui déterminera la somme totale des composants vendus pour le mois d'août. Attention cette fonction devra être utilisable pour tous les autres mois de l'année.

Question 5 :
Écrire le programme complet qui donnera le montant des bénéfices réalisés par cette entreprise pour le mois d'août.

Exercice 2. Thème calcul scientifique

Nous allons nous intéresser à des méthodes numériques recherchant le zéro d'une fonction**.**

A) Méthode de la sécante

La méthode de la sécante est un algorithme de recherche d'une racine zéro d'une fonction. Soit f cette fonction. La formule de récurrence est donnée par la relation :

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n)$$

Pour appliquer cette méthode on choisit deux pointsx_0 et $x_1$pour initialiser le processus de calcul et on arrêtel'algorithme lorsque l'écart entre deux valeurs consécutives de la suite estmajoré parune valeur fixée, par exemple 10^{-10}.

Question 6 :
Expliquer pourquoi la méthode de la sécante est une méthode dérivée de la méthode de Newton.

Question 7 :
Les deux points x_0 et x_1 peuvent-ils être choisis d'une manière quelconque ? Justifier votre réponse.

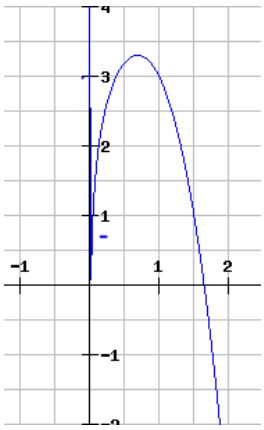
Question 8 :
Implémenter en Python la fonction sécante qui sera constituée de quatre paramètres : une fonction, deux valeurs $x_i$et la valeur de l'erreur escomptée.

$$\text{def } secante(f, x_{n_1}, x_n, erreur): \ldots$$

Question 9 :
Programmer et tester votre programme avec la fonction f définie par :

$$f : x \to \log x - x^3 + 4$$

À droite de la page vous avez une représentation partielle de cette fonction.



B) Méthode Regula Falsi

Cette deuxième méthode permet également de trouver le ou les zéros d'une fonction continue. À l'étape k on choisit deux points a_k et b_k tels que $f(a_k)$ et $f(b_k)$soient de signes contraires.Nous savons, grâce au théorème des valeurs intermédiaires, qu'il existe un point c tel que $f(c) = 0$.
On déterminera un point $c_k$le plus proche possible de c tel que :

$$c_k = a_k - \frac{a_k - b_k}{f(a_k) - f(b_k)}\, f(a_k)$$

Si $f(a_k)$ et $f(c_k)$sont de signes opposés, on choisira l'intervalle $[a_k,c_k]$ afin de continuer le processus, sinon, on choisira l'autre intervalle $[c_k, b_k]$. Si on choisira l'intervalle $[a_k,c_k]$ on posera alors $a_{k+1} = a_k$ et $b_{k+1} = c_k$. On réitère ce processus jusqu'à une valeur c_k telle que $f(c_k)$ soit aussi proche de la valeur zéro en fonction de l'erreur affichée.

Question 10 :
Quel est le principal inconvénient de la méthode Regula-Falsi ?Vous pouvez utiliser un graphique pour votre raisonnement.

Question 11 :
Implémenter la fonction « Regula_Fasi» d'une manière séquentielle.

Question 12 :
Programmer et tester votre programme toujours pour la fonction

$$f : x \to \log x - x^3 + 4$$

Problème :Gestion des "big_numbers"

L'objet de ce problème est la gestion des grands nombres, appelés encore "big_numbers". Il faut bien comprendre que la gestion des nombres entiers est liée à la spécificité du processeur de l'ordinateur. Avec un processeur 16 bits, les entiers sont codés sur un mot de 16 bits, c'est à dire qu'un entier signé sera compris entre les valeurs -2^{15} et $2^{15}-1$. Sur une machine, dite 32 bits, un entier signé sera codé sur un mot mémoire de 32 bits et sera donc compris entre -2^{31} et $2^{31}-1$. Le but, avec les "big_numbers" est de se servir d'une chaîne de caractères constituée uniquement de chiffres (0,1, 2, 3, 4, 5, 6, 7, 8, 9), de la décomposer en sous-chaînes et d'effectuer sur ces sous-chaînes de chiffres les mêmes opérations que sur les nombres.

Dans tout le problème, un entier ne sera pas géré sur un mot de 32 bits mais sur 31bits seulement, ceci pour un problème spécifiquement d'implémentation. Par conséquent un entier sera compris entre -2^{30} et $2^{30}-1$ avec $2^{30} = 1073741824$.

Dans ce problème, nous utiliserons le type *str* du langage Python pour la gestion des différentes chaînes de caractères. Par définition, il n'y a aucune contrainte sur le nombre de caractères faisant partie d'une chaîne de caractères. Les différentes chaînes de caractères seront composées uniquement de chiffres.

Dans le langage Python la taille d'un nombre entier est quelconque contrairement à la plupart des autres langages. Python utilise un système astucieux afin de faire des calculs sur des nombres aussi grands soient-ils. Les seules contraintes liées à ces calculs sont la taille mémoire de la machine et la vitesse du processeur si l'on souhaite avoir le résultat dans un temps raisonnable.Mais ce n'est pas le cas avec les autres langages. Nous allons mettre en œuvre une méthode pour permettre le calcul de grands nombres en prenant comme contrainte qu'un calcul d'un nombre entier ne peut se faire que sur un mot machine de 32 bits (en réalité 31 bits). On ne fera pas référence à la manière dont sont gérés les nombres entiers dans le langage Python

Nous cherchons à créer une fonction *"add_big_number"* effectuant la somme de grands nombres. Par exemple :

add_big_number *"1111122222333333"*
" 44555555888888"
=
"11155777778222221"

Le typage de cette fonction sera représenté par :

add_big_number : str, str → str

Les arguments de cette fonction seront donnés sous forme de chaînes de caractères et le résultat sera une chaîne de caractères.

Par hypothèse tous les nombres seront positifs. L'opérateur – unaire ne sera pas géré dans ce problème.

La méthode utilisée est la décomposition d'un nombre en sous-chaînes de N chiffres et la réalisation de morceaux de sommes afin d'obtenir le résultat. Pour les différents exemples qui

suivent, nous avons choisi arbitrairement une valeur 6 pour N. Vous devrez déterminer la véritable valeur N afin d'optimiser les différents calculs. Dans la suite du problème, nous appellerons **block** une sous-chaine composée d'un maximum de N chiffres.

Dans l'exemple précédant, les deux chaînes d'arguments peuvent se décomposer de la façon suivante :

11111	*222222*	*333333*
44	*555555*	*888888*

Les différents calculs se feront de la droite vers la gauche pour effectuer la somme des deux mots.

Le premier calcul :

```
      333333
+     888888
  ____________
1     222221
```

On conservera de ce premier calcul la valeur de la retenue qui sera prise en compte pour le second calcul et on recommencera le mécanisme sur les deux mots suivants.

Le deuxième calcul :

```
      222222
+     555555
+     1
  ____________
      777778
```

On remarque ici, qu'il n'y a pas de retenue.

Et enfin le troisième calcul :

```
          11111
+           44
  ____________
          11155
```

Le résultat final sera la concaténation des résultats partiels :

"11155777778222221"

Le résultat est constitué de 3 blocks (*11155, 777778, 222221*).

Question 13 :
Dans le cas de nombres non signés (par exemple : 123, 67, …) donner la taille maximale et la taille minimalepossible pour de tels nombres.

Question 14

Pour réaliser l'addition de deux big-numbers, déterminer la valeur N optimale. Justifier votre réponse.

Question 15 :
Implémenter la fonction subString

subString : str, int → str

subString (s,début) donne comme résultat une chaîne de longueur *N*contenant les caractères se trouvant entre l'indice *début* et l'indice *début + N-1*.
Cette fonction déclenche l'exception *IndexError* si début ouN provoque des débordements d'indices.

Pour faciliter les calculs suivants, toutes les chaînes de caractères représentant deux nombres entiers seront de même taille, c'est-à-dire composées d'un même nombre de blocs. Dans cette partie il n'y aura pas de bloc constitué uniquement de chiffres 0

Question 16 :
Écrire le code de la fonction transforme qui permet d'ajouter, au début d'une chaîne, des chiffres 0, à une chaîne composée d'un nombre inférieur à N caractères. Cette fonction sera écrite sous forme récursive. On pourra utiliser une fonction interne pour réaliser le mécanisme de récursivité.

transforme : str → str

Exemple :
```
>>>transforme("1234")# ici N vaut 6
'1234'
```

Question 17 :
Écrire le code de la fonction qui, à partir d'une chaîne donnée, déterminera le nombre de sous-chaînes composées chacune d'au maximum N caractères, c'est-à-dire le nombre de blocks.

nbSousChaines : str → int

*Ex*emple :
```
>>> nbSousChaines("1234567890")
2
```

Question 18
Écrire le code de la fonction *retenue* qui indiquera si le calcul de la somme de deux blocks partiels a ou non une valeur de retenue.

retenue : int → int

Exemple :
```
>>>retenue(1222221)
1
```

Question 19 :
Dans le cas où le calcul est avec une retenue, l'exemple précédent, il sera nécessaire de filtrer ce block de chiffres qui contient N+1 chiffre (7). Or par définition chaque bloc contient au plus N chiffres. D'autre part la retenue est utilisée dans le calcul de la somme des blocs suivants.
Écrire le code de la fonction filtre qui permet de supprimer cette retenue si elle est présente pour un nombre donné.

filtre : int → int

Exemple :
*>>> filtre(**1**222221)*
222221

Question 20 :

Écrirele code de la fonction *add_big_number*, fonction réalisant la somme de grands nombres. Le résultat sera une chaîne de caractères constituée par la concaténation de tous les calculs intermédiaires.

add_big_number :str, str → str

Exemple :
>>> add_big_number("11111111111111", "2222299999")
'000011113333411110'

On souhaite généraliser les résultats obtenus précédemment à des chaînes de caractères ne comportant pas le même nombre de blocks. Par exemple le premier nombre pourrait être constitué de 3 blocks et le second de 7 blocks.

Question 21 :
Écrire le code de la fonction*add_gene_big_number*

add_gene_big_number : str, str → str

Exemple :
>>>add_gene_big_number("11111111111111111", "999999")
'000001111111111112111110'