CONCOURS ARTS ET MÉTIERS ParisTech - ESTP - POLYTECH

Épreuve d'Informatique MP

Durée 3 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

AVERTISSEMENT

- L'épreuve est composée de 4 exercices, totalement indépendants. L'exercice 1, sur les bases de données, demande d'écrire des requêtes SQL. Les exercices suivants demandent d'écrire du code Caml.

- Un candidat pourra toujours admettre le résultat des questions qu'il n'a pas faites pour faire les questions suivantes.

La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction, la clarté et la précision** des raisonnements entreront pour une **part importante** dans **l'appréciation des copies**. En particulier, les résultats non justifiés ne seront pas pris en compte. Les candidats sont invités à encadrer les résultats de leurs calculs.

Tournez la page S.V.P.

Exercice 1

Certains prénoms sont purement masculins, ainsi Lionel et Jean-Pierre ne sont attribués qu'à des garçons. D'autres sont purement féminins, comme Angélique et Delphine, qui ne sont attribués qu'à des filles. Enfin, d'autres prénoms sont donnés, avec la même orthographe, à des garçons et à des filles, comme Andréa, Alix et Dominique. Ces prénoms sont dits *épicènes*.

Le taux de féminité d'un prénom P est la proportion de filles appelées P à la naissance sur le nombre total de bébés prénommés P. Par exemple, le taux de féminité f d'Alix, donné à 8217 filles françaises et 2360 garçons est donné par la formule

$$f = \frac{8217}{2360 + 8217} \approx 0.777 = 77.7\%.$$

Ce taux de féminité est utilisé dans le cadre d'études sociologiques pour "deviner" le sexe quand seul le prénom est connu. Par exemple, il fut utilisé par la sociologue Valérie Carasco, dans une étude pour le ministère de la justice publiée en octobre 2007, pour attribuer un genre aux PACS : féminin (deux femmes) ou masculin ou mixte.

Dans cet exercice, nous disposons d'une base de données contenant une table `baseprenoms` ayant la forme suivante :

prenom	nombre	sexe	annee	departement
Manon	19.0	F	1983	Bouches-du-Rhône
Zakaria	24.0	M	2006	Hauts-de-Seine
Andrea	23.0	F	2001	Gironde
Andrea	30.0	M	2004	Alpes-Maritimes
⋮	⋮	⋮	⋮	⋮

Cette table indique pour chaque année, chaque département, chaque prénom et chaque sexe, le nombre de bébés nés avec ce prénom. Ainsi la troisième ligne signifie qu'en 2001, en Gironde, 23 bébés filles furent prénommées "Andréa".

Dans cet exercice, chaque question demande d'écrire une requête, et est suivie de quelques lignes retournées par la requête. Lorsque le résultat d'une requête est sauvegardé sous un nom, il peut être utilisé dans une autre requête comme n'importe quelle table.

1. Écrire une requête donnant la table des prénoms féminins et le nombre de filles nées avec ce prénom. Écrire une requête donnant la table des prénoms masculins ainsi que le nombre de garçons nés avec ce prénom.

prenom	nombreF
Alix	8217.0
Charlie	162.0
Julie	171878.0
⋮	⋮

prenom	nombreM
Alix	2360.0
Charlie	2909.0
Jean-Claude	124137.0
⋮	⋮

On supposera par la suite que les résultats de ces deux requêtes sont sauvegardés sous les noms respectifs `feminin` et `masculin`.

2. Écrire une requête donnant la table des prénoms épicènes avec le nombre de filles et le nombre de garçons nés avec ce prénom ainsi que le taux de féminité.

prenom	nombreF	nombreM	TauxF
Alix	8217.0	2360.0	0.777
Charlie	162.0	2909.0	0.053
Dominique	219359.0	157761.0	0.418
⋮	⋮	⋮	⋮

On supposera par la suite que cette requête est sauvegardé sous le nom `epicene`.

3. Écrire une requête renvoyant la table des prénoms exclusivement féminins. Écrire une requête renvoyant la table des prénoms exclusivement masculins.

prenom
Angélique
Delphine
⋮

prenom
Lionel
Jean-Pierre
⋮

On supposera par la suite que les résultats de ces deux requêtes sont sauvegardés sous les noms respectifs `prenomfeminin` et `prenommasculin`.

4. Écrire une requête renvoyant la liste des prénoms avec leur taux de féminité.

prenom	tauxF
Alix	0.777
Angelique	1.0
Lionel	0.0
⋮	⋮

Exercice 2

On considère le programme suivant, écrit en Caml :

```
let rec pow a k = if k=0 then 1 else a* pow a (k-1);;
```

`pow a k` calcule `a` à la puissance `k`.

1. Quelle est la complexité en temps de `pow` ?
2. Que dire de l'occupation de la mémoire lors de l'exécution de `pow` ? Estimer la complexité en mémoire (on dit aussi en espace) de `pow`.

Tournez la page S.V.P.

Dans la suite de cet exercice, on utilise une version de Python travaillant sur des entiers illimités, et une version de Caml travaillant sur des entiers de 31 bits. Les entiers représentables en Caml vont de -2^{30} à $2^{30}-1$ inclus. Lorsqu'une opération arithmétique dans $\mathbb{Z}$ renvoie un entier k qui n'est pas représentable en Caml, alors Caml renvoie l'unique entier représentable congru à k modulo 2^{31}.

`90000*80000;;` renvoie `757549056` car $90000 \times 80000 = 7200000000 \equiv 757549056 \left[2^{31}\right]$ et $-2^{30} \leqslant 757549056 < 2^{30}$. De même `99000*80000;;` renvoie `-669934592` car $99000 \times 80000 = 7920000000 \equiv -669934592 \left[2^{31}\right]$ et $-2^{30} \leqslant -669934592 < 2^{30}$.

3. Quelle valeur renvoie `pow 2 3` ? Quelle valeur renvoie `pow 2 42` ?

Considérons les fonctions suivantes écrites en Python et Caml pour un entier `n` > 0 :

Python

```
def f(n):
    k=0
    while 2**k<=n:
        k=k+1
    return k
```

Caml

```
let f n =
    let k=ref 0 in
    while pow 2 !k <= n do
        k:= !k+1 done;
    !k;;
```

4. Décrire l'exécution de `f(3)` en Python.
5. Que calcule la `f(n)` en Python pour `n` un entier strictement positif ?
6. Dans cette question, nous supposons que `n` prend des valeurs entières, strictement positives, et représentables en Caml.
   (a) Pour quelles valeurs de `n` les fonctions `f` en Python et en Caml renvoient le même résultat ?
   (b) Que se passe-t-il, pour les autres valeurs de `n`, lors du calcul de `f n` en Caml ?
7. Réécrire la fonction `f` de Caml pour qu'elle renvoie le même résultat que la fonction Python sur tous les entiers strictement positifs représentables en Caml.

Exercice 3

On suppose disposer d'une structure impérative de dictionnaire en Caml de type `('a,'b) dict` avec les primitives suivantes :

Primitive	Type	Description
`new`	`unit -> ('a,'b) dict`	Crée un nouveau dictionnaire vide
`add`	`('a,'b) dict -> 'a -> 'b -> unit`	`add d cl val` associe, dans le dictionnaire `d`, la clef `cl` à la valeur `val`
`find`	`('a,'b) dict -> 'a -> 'b`	`find d cl` lit, dans le dictionnaire `d`, la valeur associée à la clef `cl`
`keys`	`('a,'b) dict -> 'a list`	`keys d` renvoie la liste (dans un ordre arbitraire) des clefs du dictionnaire `d`

`'a` est le type des clefs, et `'b` le type des valeurs associées aux clefs.

On suppose disposer d'une structure persistante d'ensemble d'entiers de type `set` avec les primitives suivantes :

Primitive	Type	Description
`empty`	`set`	L'ensemble vide
`union`	`set -> set -> set`	L'union de 2 ensembles
`inter`	`set -> set -> set`	L'intersection de 2 ensembles
`card`	`set -> int`	Le cardinal d'un ensemble
`equal`	`set -> set -> bool`	Teste l'égalité de 2 ensembles
`mem`	`int -> set -> bool`	`mem i s` renvoie `true` si l'entier `i` appartient à l'ensemble `s` et `false` sinon
`singleton`	`int -> set`	`singleton i` renvoie l'ensemble à un élément ne contenant que `i`

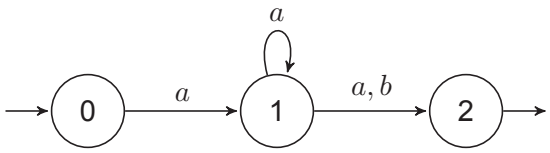
On représente un automate non-déterministe en Caml par le type suivant

```
type auto = {etats : set; init: set;
        trans: (int * char, set) dict ; final: set};;
```

Étant donné un automate `m`,

- `m.etats` représente l'ensemble des états de `m`,
- `m.init` représente l'ensemble des états initiaux de `m` ;
- `m.final` l'ensemble de ses états finaux,
- `m.trans` sa fonction de transition qui à chaque couple `q,x` associe l'ensemble des états accessibles à partir de l'état `q` en lisant le caractère `x`.

Par exemple, considérons l'automate $\mathcal{M}_1$ suivant :



Si `m1` représente l'automate $\mathcal{M}_1$ en Caml alors `m1.final` est l'ensemble $\{2\}$ et `find m1.trans (1,`a`)` est l'ensemble $\{1; 2\}$.

Un automate déterministe est un automate non-déterministe ayant un unique état initial et tel que pour tout état q et toute lettre a, en lisant a à partir de l'état q on peut aller dans au plus un état.

1. L'automate $\mathcal{M}_1$ est-il déterministe ?
2. Écrire une fonction `max_card : ('a, set) dict -> int` qui étant donné un dictionnaire d'ensembles, renvoie le cardinal maximal des ensembles stockés dans le dictionnaire.

Tournez la page S.V.P.

3. Écrire une fonction `est_deterministe : auto -> bool` qui renvoie `true` si l'automate donné en argument est déterministe et `false` sinon.

4. Considérons le code incomplet suivant :

```
let etats_suivants m s x =                                    1
let rec parcours_clefs clefs acc = match clefs with           2
        | [] -> acc                                           3
        | (etat,y)::r -> if  ...............                  4
                 then  ...............                        5
                 else  ...............                        6
in parcours_clefs (keys m.trans) empty;;                      7
```

   Compléter le code de sorte que `etats_suivants m s x` renvoie l'ensemble des états accessibles à partir d'un état de l'ensemble `s` en lisant `x` dans l'automate `m`.

5. Écrire une fonction `reconnu : auto -> string -> bool` qui, étant donné un automate et une chaîne de caractères, renvoie `true` si l'automate reconnaît la chaîne et `false` sinon.

6. Si au lieu d'une structure impérative de dictionnaire nous avions une structure persistante de dictionnaire, quel serait le type de la primitive `add` ? Si nous avions une structure impérative d'ensemble d'entiers et non pas une structure persistante, pourquoi le type de `empty` devrait-il être changé ?

Exercice 4

Nous souhaitons classer les fromages français dans un arbre : les fromages les plus proches seront sur des branches "voisines" de l'arbre. Dans ce but est défini un type d'arbre étiqueté : `type arbre = Fromage of int | N of arbre * arbre * float;;`

Pour chaque fromage nous disposons d'un ensemble de mesures : apport énergétique, teneur en calcium, en protéines, …

Nom	Apport énergétique (en kJ/100g)	Calcium (en mg/100g)	…
Camembert	1150	333	…
Maroilles	348	335	…
Roquefort	374	601	…
⋮	⋮	⋮	⋮

Le tableau précédent est représenté en Caml par une matrice globale `fromages` de flottants. Par exemple, `fromages.(0).(1)` donne `333.0`, la teneur en calcium (colonne numéro 1) du camembert (fromage numéro 0). Chaque fromage f est alors représenté par une ligne $[|f_0; \ldots; f_q|]$ de la matrice `fromages`.

Étant donné un tableau global `ponderation`$= [|\alpha_0; \ldots; \alpha_q|]$ de flottants strictement positifs on définit la distance $d(a, b)$ entre deux fromages $a = [|x_0; \ldots; x_q|]$ et $b =$

$[|y_0; \ldots; y_q|]$ par $d(a,b) = \sum_{k=0}^{q} \alpha_k |x_k - y_k|$, puis on définit la distance $d(A,B)$ entre deux ensembles de fromages A et B par

$$d(A,B) = \min_{a \in A \text{ et } b \in B} d(a,b)$$

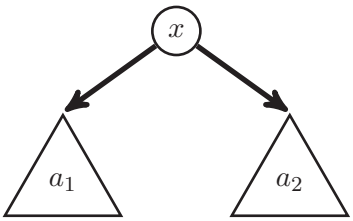
Enfin, on définit la distance entre deux arbres a_1 et a_2 comme la distance entre l'ensemble des fromages présents dans a_1 et l'ensemble des fromages présents dans a_2.

1. Écrire une fonction `distance : int -> int -> float`.
   `distance i j` calcule la distance entre les fromages de numéros `i` et `j`.

2. Définir en Caml une matrice globale `dist`, telle que, pour tous fromages `i` et `j`, `dist.(i).(j)` est égal à la distance entre les fromages `i` et `j`.

3. Écrire une fonction `dist_arbr : arbre -> arbre -> float` qui étant donné deux arbres de fromages, calcule la distance entre ces deux arbres.

4. La fusion de deux arbres a_1 et a_2 est un arbre de la forme ci-après où la racine est étiquetée par x, la distance entre a_1 et a_2.



   On appelle *forêt* une liste d'arbres. Un dendrogramme est un arbre construit de la manière suivante.

   (a) On crée une forêt contenant un arbre de la forme `Fromage(i)` par fromage.
   (b) Tant qu'il y a plusieurs arbres dans la forêt, on cherche deux arbres séparés par la plus petite distance possible, puis on les fusionne.
   (c) Lorsqu'il ne reste plus qu'un seul arbre, on renvoie celui-là.

   Écrire une fonction `dendrogramme : unit -> arbre` qui crée le dendrogramme de tous les fromages référencés dans la matrice globale `fromages`.