DS 1 : jeudi 5 novembre

Nom : Classe :

2h sans calculatrice

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Exercice 1 *(SQL : Station de ski).*
Nous avons à notre disposition la base de données d'une station de ski composée de tables dont la structure est la suivante :

Domaines	
id_domaine	nom

La table « Domaines » contient les informations sur les domaines skiables pour lesquels un forfait peut être acheté.

Bornes				
id	id_borne	nom	id_domaine	temps

La table « Bornes » contient les informations utiles de chaque borne permettant l'accès à une remontée mécanique, l'information importante étant de quel domaine elle fait partie. Une même borne peut appartenir à plusieurs domaines. Le temps correspond au temps minimum nécessaire pour la remontée concernée.

Bureaux		
id_bureau	nom	type

La table « Bureaux » contient les noms des différents bureaux où peuvent être achetés des forfaits, et leurs types (bornes automatiques, caisse).

Forfaits					
id_forfait	id_domaine	id_bureau	date	heure	validité

La table « Forfaits » contient tous les forfaits achetés, domaine de validité, bureau d'achat, date et heure de début de validité et durée de validité en heures. Tous les forfaits sont initialisés pour un début de validité à 00h00. Pour les forfaits 5h, l'heure sera mise à jour lors du premier passage dans une borne.

Clients			
id_client	nom	prenom	id_forfait

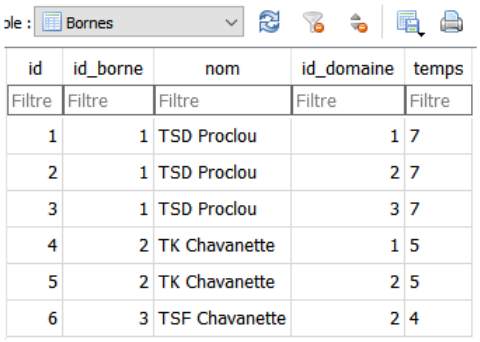
La table « Clients » contient pour chaque client, son nom et son prénom, et l'id du forfait qu'il a acheté.

Passages					
id_passage	id_borne	date	heure	id_forfait	passage

La table « Passages » répertorie, à chaque tentative de passer à une borne, l'id de la borne, la date et heure de passage du forfait, l'id du forfait associé et le résultat (passage autorisé (1) ou non(0)).

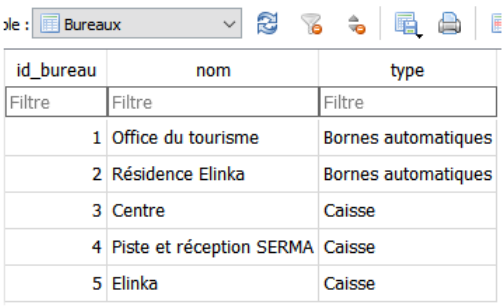
La base de données relationnelle est donc :
— Domaines (id_domaine : Integer, nom : Text)
— Bornes (id : Integer, id_borne : Integer, nom : Text, id_domaine : Integer, temps : Integer)
— Bureaux (id_bureau : Integer, nom : Text, type : Text)
— Forfaits (id_forfait : Integer, id_domaine : Integer, id_bureau : Integer, date : Date, heure : Datetime, validité : Integer)
— Clients (id_client : Integer, nom : Text, prenom : Text, id_forfait : : Integer)
— Passages (id_passage : Integer, id_borne : Integer, date : Date, heure : Datetime, id_forfait : Integer, passage : Integer)

Voici quelques captures d'écran issus du logiciel DB Browser :



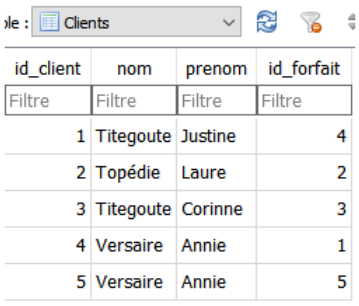
Bornes

id	id_borne	nom	id_domaine	temps
Filtre	Filtre	Filtre	Filtre	Filtre
1	1	TSD Proclou	1	7
2	1	TSD Proclou	2	7
3	1	TSD Proclou	3	7
4	2	TK Chavanette	1	5
5	2	TK Chavanette	2	5
6	3	TSF Chavanette	2	4



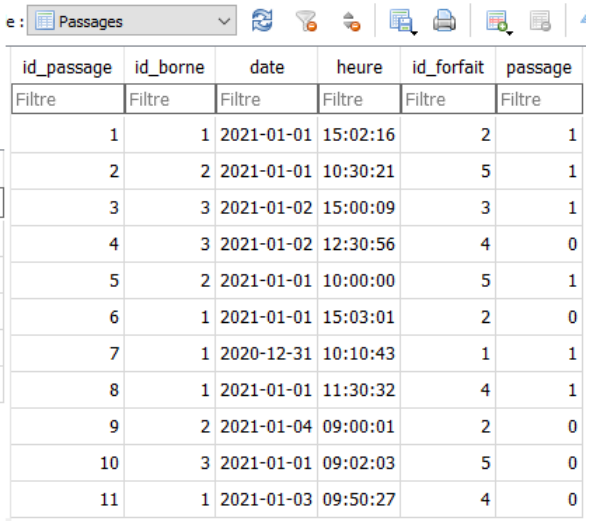
Bureaux

id_bureau	nom	type
Filtre	Filtre	Filtre
1	Office du tourisme	Bornes automatiques
2	Résidence Elinka	Bornes automatiques
3	Centre	Caisse
4	Piste et réception SERMA	Caisse
5	Elinka	Caisse



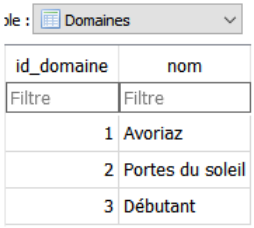
Clients

id_client	nom	prenom	id_forfait
Filtre	Filtre	Filtre	Filtre
1	Titegoute	Justine	4
2	Topédie	Laure	2
3	Titegoute	Corinne	3
4	Versaire	Annie	1
5	Versaire	Annie	5



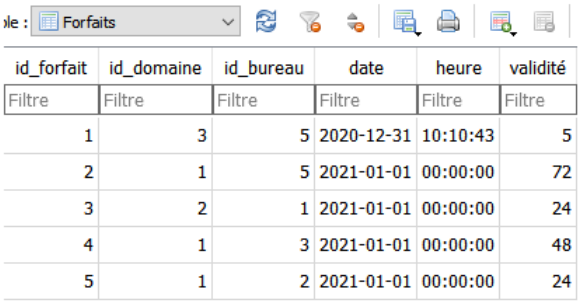
Passages

id_passage	id_borne	date	heure	id_forfait	passage
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	2021-01-01	15:02:16	2	1
2	2	2021-01-01	10:30:21	5	1
3	3	2021-01-02	15:00:09	3	1
4	3	2021-01-02	12:30:56	4	0
5	2	2021-01-01	10:00:00	5	1
6	1	2021-01-01	15:03:01	2	0
7	1	2020-12-31	10:10:43	1	1
8	1	2021-01-01	11:30:32	4	1
9	2	2021-01-04	09:00:01	2	0
10	3	2021-01-01	09:02:03	5	0
11	1	2021-01-03	09:50:27	4	0



Domaines

id_domaine	nom
Filtre	Filtre
1	Avoriaz
2	Portes du soleil
3	Débutant



Forfaits

id_forfait	id_domaine	id_bureau	date	heure	validité
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	3	5	2020-12-31	10:10:43	5
2	1	5	2021-01-01	00:00:00	72
3	2	1	2021-01-01	00:00:00	24
4	1	3	2021-01-01	00:00:00	48
5	1	2	2021-01-01	00:00:00	24

Réaliser les requêtes suivantes :

1° Lister les noms des domaines de la station

2° Lister les noms des bornes sans doublons

3° Compter le nombre de domaines

4° Trouver les id des passages autorisés du forfait 4

5° Lister les clients (nom, prénom) ayant acheté plusieurs forfaits

6° Sachant qu'une même borne peut appartenir à plusieurs domaines, trouver les noms des bornes qui ne sont pas dans le domaine n° 3 (débutant)

7° Donner les couples nom/prénom des personnes dont la validité du forfait commence le 31/12/2020

8° Lister les dates et heures des passages de forfaits « Débutant » (aucun pré-traitement humain)

9° Trouver la liste des heures distinctes de passage de forfaits au « TK Chavanette »

10° Trouver la liste des id des passages ayant respecté le domaine de leur forfait

11° En déduire la liste des id des passages n'ayant pas respecté le domaine de leur forfait (Bonus)

Exercice 2 *(E3A PSI, 2018)*.

1° Proposer une fonction python `maxi` prenant en argument une liste d'entiers naturels `L` et renvoyant le maximum des entiers de cette liste.
On n'utilisera pas de fonction spécifique de python déterminant ce maximum.

2° Écrire une fonction `ind` prenant en argument une liste d'entiers naturels `L` et renvoyant la liste des indices `[i_1,...,i_r]` avec `i_1<...<i_r` telle que pour tout $k \in [\![1, r]\!]$, `L[i_k]` soit non nul.
Par exemple si `L = [0,1,3,0,7]`, alors `ind(L)` renvoie `[1,2,4]`.

3° Écrire une fonction `nb_oc` prenant comme argument une liste d'entiers naturels `L` et renvoyant la liste `T` de longueur `M = maxi(L)+1` où, pour tout $i \in [\![0, M]\!]$, `T[i]` est le nombre d'occurrences de l'entier `i` dans la liste `L`.
Par exemple, si `L = [3,1,4,1,5]`, alors `T = [0,2,0,1,1,1]`.
On pourra utiliser la fonction `maxi`.

4° Soit `L` une liste d'entiers naturels.

4°.1. Déterminer le nombre de fois, noté `n`, où la liste `L` est parcourue lors de l'exécution de `nb_oc(L)`.

4°.2. On veut que ce nombre `n` soit indépendant de `M = maxi(L)+1`.
Si ce n'est pas le cas, modifier la fonction `nb_oc` afin de respecter cette condition.

5° Soit A une liste d'entiers naturels. On définit la suite de Robinson $(L_n)_{n\in\mathbb{N}}$ associée à la suite A par récurrence comme suit :

— $L_0 = A$.
— Si L_n est construite, alors :
  — on détermine $T_n = \texttt{nb_oc}(L_n)$.
  — on détermine $I_n = \texttt{ind}(T_n)$.
  — si $I_n = [i_1, \dots, i_r]$, alors $L_{n+1} = \big[T[i_r], i_r, \dots, T[i_1], i_1\big]$.

Par exemple si $A = [4, 4, 1, 2]$, alors :
— $L_0 = [4, 4, 1, 2]$
— $L_1 = [2, 4, 1, 2, 1, 1]$ (il y a deux « 4 », un « 2 » et un « 1 » dans la liste L_0)
— $L_2 = [1, 4, 2, 2, 3, 1]$ (il y a un « 4 », deux « 2 » et trois « 1 » dans la liste L_1)

5°.1. On donne $A = [2, 0, 4, 1, 3, 3, 2, 3, 1, 1]$. Déterminer L_3 et L_{2018}.

5°.2. On donne $B = [2, 4, 1, 1, 1, 2]$. Si l'on suppose que $L_1 = B$, donner toutes les solutions possibles pour L_0.

5°.3. On donne $C = [2, 4, 1, 0]$. Si l'on suppose que $L_1 = C$, donner toutes les solutions possibles pour L_0.

5°.4. Proposer alors une fonction `rob(A,n)` qui prend en arguments une liste d'entiers naturels `A` et un entier naturel `n` et qui renvoie l'élément L_n de la suite de Robinson associée à `A`.

Exercice 3.

L'objectif est de déterminer la plus petite somme d'éléments consécutif dans une liste.

Par exemple, si `L = [1, -3, 4, -2, 1, -5, 7]` alors la plus petite somme d'éléments consécutifs est $-2+1-5=-6$.

1° Première méthode : force brut

(a) Écrire une fonction d'entête `somme(L,i,j)` qui retourne la somme des éléments de `L` entre les indices `i` et `j`. Par exemple, `somme([1, -3, 4, -2, 1, -5, 7],2,4)` doit renvoyer `3`.

(b) Écrire une fonction d'entête `somme_min1(L)` qui retourne la plus petite somme d'éléments consécutifs de `L`

(c) Quel est la complexité de `somme_min1` en fonction de la taille n de la liste.
On commencera par donner le nombre d'additions de `somme(L,i,j)` pour en déduire celui de `somme(L)` (et passer dés que possible aux grand O pour limiter les calculs).

2° Deuxième méthode : en gardant les sommes partielles.

(a) Écrire une fonction `cumsum` prenant en entrée une liste, et renvoyant la liste des sommes cumulées (depuis le premier terme). La complexité devra être en $O(n)$ où n est le nombre d'éléments dans la liste.
Par exemple `cumsum([1,2,3,4])` doit retourner `[1,3,6,10]`.

(b) Si on exécute `S=cumsum(L)`, alors `S[i]` contient $\sum_{k=0}^{i} \texttt{L[k]}$. Exprimer $\sum_{k=i}^{j} \texttt{L[k]}$ à l'aide de `S`.

(c) En déduire une fonction d'entête `somme_min2(L)` qui retourne la plus petite somme d'éléments consécutifs de `L` (qui utilisera donc `cumsum` au lieu de `somme`).

(d) Quel est sa complexité ? (on ne détaillera pas)

3° Troisième méthode : algorithme de Kadane.
On admet que l'algorithme suivant (écrit en pseudo-code) permet de calculer, dans `s`, la plus petite somme d'éléments consécutifs de `L` :
— `s` reçoit la valeur `0`.
— `s_cur` reçoit la valeur `0`.
— **Pour** chaque élément `e` de `L`, **faire** :
  — `s_cur` reçoit la valeur `s_cur+e`.
  — **Si** `e<s_cur` **alors** `s_cur` reçoit la valeur `e` **Fin du si**.
  — **Si** `s_cur<s` **alors** `s` reçoit la valeur `s_cur` **Fin du si**.
— **Fin du pour**

(a) Traduire (on dit aussi implémenter) cet algorithme en Python sous la forme d'une fonction `somme_min3(L)` qui retournera la valeur finale de `s`.

(b) Quel est la complexité de ce dernier algorithme ? (on ne détaillera pas)