

Concours Blanc - Epreuve d'informatique

⚠ Consignes

- Les programmes demandés doivent être écrits en C ou en OCaml suivant l'exercice. Dans le cas du C, on suppose que les bibliothèques standards usuelles (`<stdio.h>`, `<stdlib.h>`, `<stdbool.h>`) sont déjà importées.
- On pourra toujours librement utiliser une fonction demandée à une question précédente même si cette question n'a pas été traitée.
- Veillez à présenter vos idées et vos réponses partielles même si vous ne trouvez pas la solution complète à une question.
- La clarté et la lisibilité de la rédaction et des programmes sont des éléments de notation.

□ Exercice 1 : Base de données et SQL

🎓 CAPES NSI 2021, épreuve 1

On s'intéresse dans cette partie à un site Internet d'échange de supports de cours entre enseignants de MP2I/MPI. Chaque personne désirant proposer ou récupérer du contenu doit commencer par se créer un compte sur ce site et peut ensuite accéder à du contenu ou en proposer.

Ce site repose sur une base de données contenant en particulier une table, nommée **ressources**. Elle possède un enregistrement par document téléversé sur le site. Ses attributs sont :

- **id**, un identifiant numérique, unique pour chaque ressource ;
- **owner**, le pseudo de la personne ayant créé la ressource ;
- **annee**, l'année de publication de la ressource ;
- **titre**, une chaîne de caractères décrivant la ressource ;
- **type**, chaîne de caractères pouvant être cours, ds, tp ou td.

Voici un extrait de cette table :

id	owner	annee	titre	type
4	dknuth	2020	Machine à décalage	cours
13	alovelace	2022	Intelligence artificielle	td
...

- Q1**– Écrire une requête SQL permettant de connaître tous les titres des ressources déposées par « **jclarke** » classées par année de publication croissante.

```
SELECT titre
FROM ressources
WHERE owner = "jclarke"
ORDER BY annee ASC;
```

- Q2**– Écrire une requête SQL permettant de connaître le nombre total de ressources de type cours présentes sur le site.

```
SELECT COUNT(*)
FROM ressources
GROUP BY type HAVING type = "cours";
```

- Q3**– Que fait la requête suivante : ?

```
SELECT R.owner
FROM Ressources AS R
WHERE R.type = 'td'
GROUP BY R.owner
ORDER BY COUNT(*) DESC
LIMIT 3
```

Cette requête permet d'afficher les trois premiers propriétaires de ressources ayant posté le plus de ressources de type td

❑ **Exercice 2** : Logique et calcul des propositions

CCINP 2018 - MP

Vous avez été sélectionné(e) pour participer au jeu "Cherchez les Clés du Paradis (CCP)". Le jeu se déroule en trois épreuves, au cours desquelles vous devez collecter des clés vertes. A l'issue de chacune d'entre elles, vous passez à l'épreuve suivante en cas de succès et êtes éliminé(e) en cas d'échec.

■ **Partie I** : Première épreuve

Jean-Pierre Pendule, le célèbre animateur, vous accueille pour la première épreuve. Il vous explique la règle du jeu. Devant vous, deux boîtes et sur chacune d'entre elles une inscription. Chacune des boîtes contient soit une clé verte, soit une clé rouge. Vous devez choisir l'une des boîtes : si le résultat est une clé rouge, alors vous quittez le jeu, si c'est une clé verte vous êtes qualifié(e) pour l'épreuve suivante.

Jean-Pierre Pendule dévoile les inscriptions sur chacune des boîtes et vous affirme qu'elles sont soit vraies toutes les deux, soit fausses toutes les deux :

- sur la boîte 1, il est écrit « Une au moins des deux boîtes contient une clé verte » ;
- sur la boîte 2, il est écrit « Il y a une clé rouge dans l'autre boîte ».

Dans toute cette partie, on note P_i la proposition affirmant qu'il y a une clé verte dans la boîte i .

Q4– Donner une formule de la logique des propositions représentant la phrase écrite sur la boîte 1.

La formule logique associée à la phrase de la boîte 1 est $P_1 \vee P_2$

Q5– Donner de même une formule de la logique des propositions pour l'inscription de la boîte 2.

La formule logique associée à la phrase de la boîte 2 est $\neg P_1$

Q6– Donner une formule représentant l'affirmation de l'animateur. Simplifier cette formule de sorte à n'obtenir qu'une seule occurrence de chaque P_i .

L'affirmation de l'animateur se traduit par $Q = ((P_1 \vee P_2) \wedge \neg P_1) \vee (\neg(P_1 \vee P_2) \wedge \neg \neg P_1)$
 $Q = ((P_1 \wedge \neg P_1) \vee P_2 \wedge \neg P_1) \vee (\neg P_1 \wedge \neg P_2 \wedge P_1)$
 $= P_2 \wedge \neg P_1$

Q7– Quel choix devez-vous faire pour continuer le jeu à coup sûr ?

D'après la question précédente, la clé verte se trouve dans la boîte 2, il faut donc choisir cette boîte.

■ **Partie II** : Deuxième épreuve

Bravo, vous avez obtenu la première clé verte. Jean-Pierre Pendule vous félicite et vous annonce que cette première épreuve n'était qu'une mise en jambe. Avec les mêmes règles du jeu, l'animateur vous propose alors deux nouvelles boîtes portant les inscriptions suivantes :

- sur la boîte 1, il est écrit « Il y a une clé rouge dans cette boîte, ou bien il y a une clé verte dans la boîte 2 » ;
- sur la boîte 2, il est écrit « Il y a une clé verte dans la boîte 1 ».

Q8– Donner une formule de la logique des propositions pour chaque affirmation.

Q9– Sachant qu'encore une fois les deux affirmations sont soit vraies toutes les deux, soit fausses toutes les deux, donner le contenu de chaque boîte. En déduire votre choix pour remporter la deuxième clé verte.

■ Partie III : Troisième épreuve

Le suspens est à son comble, vous voici arrivé(e) à la dernière épreuve. A votre grande surprise, Jean-Pierre Pendule vous dévoile une troisième boîte et vous explique les règles du jeu. Dans une des boîtes se cache la clé qui vous permet de remporter la victoire finale. Dans une autre boîte se cache une clé rouge qui vous fait tout perdre. La dernière boîte est vide. Encore une fois, chacune des boîtes porte une inscription :

- sur la boîte 1, il est écrit « La boîte 3 est vide » ;
- sur la boîte 2, il est écrit « La clé rouge est dans la boîte 1 » ;
- sur la boîte 3, il est écrit « Cette boîte est vide ».

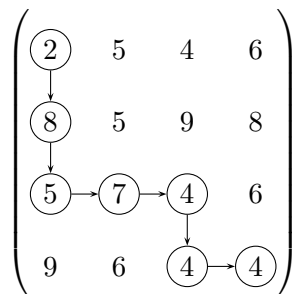
L'animateur affirme que l'inscription portée sur la boîte contenant la clé verte est vraie, celle portée par la boîte contenant la clé rouge est fausse. L'inscription affichée sur la boîte vide est aussi vraie.

- Q10**– Donner une formule de la logique des propositions pour chaque inscription.
- Q11**– Donner une formule de la logique des propositions synthétisant l'information que vous a apportée l'animateur.
- Q12**– En supposant que la clé verte est dans la boîte 2, montrer que l'on aboutit à une incohérence.
- Q13**– Donner alors la composition des trois boîtes.

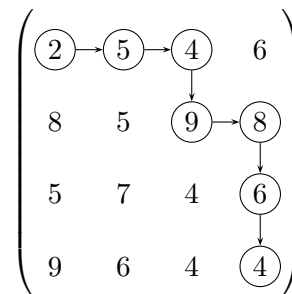
□ Exercice 3 : Traversée maximale d'une matrice carrée

Les fonctions demandées dans cet exercice doivent être écrites en OCaml. On rappelle que qu'en OCaml, on peut convertir une liste en tableau à l'aide de `Array.of_list` par exemple `Array.of_list [6; 7; 42;]` renvoie `[| 6; 7; 42; |]`

On considère une matrice à n lignes et n colonnes notée M à coefficients dans \mathbb{N} , et dont le coefficient situé à la ligne i et à la colonne j avec $0 \leq i \leq n-1$ et $0 \leq j \leq n-1$ sera noté $M_{i,j}$. On s'intéresse aux chemins depuis la première valeur en haut à gauche ($M_{0,0}$) jusqu'à la dernière en bas à droite ($M_{n-1,n-1}$) qui n'utilisent que les déplacements vers la droite (\rightarrow) ou vers le bas (\downarrow). Voici deux exemples de tels chemins dans une même matrice A de 4 lignes et 4 colonnes :



Un exemple C_1 de chemin dans A



Un exemple C_2 de chemin dans A

On appelle *somme d'un chemin* dans une matrice M , la somme des coefficients $M_{i,j}$ rencontrés en suivant ce chemin. Dans les exemples ci-dessus, le chemin C_1 a une somme de 34 et le chemin C_2 a une somme de 38. Le but du problème est d'étudier divers algorithmes qui cherchent à déterminer un chemin dont la somme est maximale.

■ Partie I : Un premier algorithme naïf

On propose l'algorithme suivant afin de résoudre ce problème : si les deux directions bas ou droite sont possibles on choisit celle qui mène vers le coefficient de matrice le plus élevé, en cas d'égalité, on choisit (arbitrairement) la direction bas. Dans l'exemple ci-dessus à partir de $M_{0,0} = 2$ on peut aller vers à droite vers $M_{0,1} = 5$ ou en bas vers $M_{1,0} = 8$ et on choisit donc d'aller en bas puisque $8 > 5$. Une fois arrivé en $M_{1,0}$ les deux directions possibles mènent vers des coefficients identiques (5 des deux côtés) et donc, on va vers le bas. Enfin, lorsqu'une seule direction est possible (parce qu'on a atteint la dernière colonne ou la dernière ligne), c'est cette direction qui est choisie.

- Q14**– Poursuivre le déroulement de cet algorithme sur la matrice A donnée en exemple ci-dessus, dessiner le chemin obtenu et donner sa somme.
- Q15**– Dans quelle famille d'algorithme peut-on classer cet algorithme ? Justifier.
- Q16**– Donner la complexité de cet algorithme.

- Q17**– Montrer en exhibant un exemple dans le cas $n = 4$ que cet algorithme ne fournit pas toujours un chemin de somme maximale.
- Q18**– Montrer que pour une matrice de taille n , un chemin de $M_{0,0}$ à $M_{n-1,n-1}$ est constitué de $2n - 2$ directions.
- Q19**– On définit le type :

```
1 type direction = Droite | Bas;;
```

Ecrire une fonction `choix : int array array -> int -> int -> direction` qui prend en argument une matrice d'entiers `M`, une ligne `lig` et une colonne `col` et renvoie `Bas` ou `Droite` suivant le choix effectué par l'algorithme décrit ci-dessus à partir de la position située à la ligne `lig` à la colonne `col` de la matrice `M`.

- Q20**– Ecrire une fonction `naïf int array array -> direction array` qui prend en entrée une matrice d'entiers `M` et renvoie le chemin fournit par l'algorithme naïf décrit ci-dessus.
- Q21**– Ecrire une fonction `somme int array array -> direction array -> int` qui prend en entrée une matrice d'entiers `M`, un chemin dans cette matrice et renvoie la somme de ce chemin.

■ Partie II : Résolution exacte par force brute

On propose l'algorithme par force brute qui consiste à énumérer tous les chemins possibles de $M_{0,0}$ vers $M_{n-1,n-1}$ à calculer leur somme puis à renvoyer la somme maximale atteignable. Afin de décrire un chemin on le fait correspondre à un tableau de $n - 1$ entiers compris entre 1 et $2n - 2$, qui indique la position des `Bas` dans le chemin. Par exemple pour $n = 3$, le chemin `[| Bas; Droite; Bas; Droite |]` sera représenté par `[| 1; 3 |]` car les `Bas` se trouvent en première et troisième position du tableau. De la même façon, le chemin C_1 donné en exemple en début d'énoncé dans le cas $n = 4$ est `[| Bas; Bas; Droite; Droite; Bas; Droite |]` et sera donc représenté par `[| 1; 2; 5 |]`.

- Q22**– A quel chemin correspond le tableau d'entiers `[|3; 5; 6|]` ?
- Q23**– Ecrire une fonction `cree_chemin int array -> direction array` qui prend en argument un tableau d'entiers et renvoie le chemin correspondant. Par exemple, `cree_chemin [| 4; 1 |]` renvoie le tableau `[| Bas; Droite; Droite; Bas |]`.
- Q24**– Ecrire en OCaml une fonction `entiers : int -> int list` qui prend en entrée un entier `n` et renvoie la liste des entiers de `n` à 1. Par exemple `entiers 4` renvoie la liste `[4; 3; 2; 1]`.
- Q25**– On souhaite maintenant énumérer les tableaux d'entiers représentant les chemins. Ecrire en OCaml une fonction `chemins : int -> int array list` qui prend en argument un entier `n` et renvoie la liste des tableaux d'entiers représentant les chemins possibles. Par exemple `chemins 2` renvoie `[[| 4; 3 |]; [| 4; 2 |]; [| 4; 1 |]; [| 3; 2 |]; [| 3; 1 |]; [| 2; 1 |]]`.
- Q26**– En déduire une fonction `force_brute : int array array -> int` qui prend en entrée une matrice d'entiers et renvoie la somme maximale atteignable en traversant cette matrice.
- Q27**– Montrer que pour une matrice M de n lignes et n colonnes, il existe $\binom{2n-2}{n-1}$ chemins possibles de $M_{0,0}$ vers $M_{n-1,n-1}$.
- Q28**– En déduire la complexité de l'algorithme par force brute en admettant que le calcul de la somme d'un chemin donné est en $O(n)$ où n est la longueur de ce chemin. On pourra utiliser la formule de Stirling : $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$.

■ Partie III : Résolution par programmation dynamique

Etant donné une matrice carré M de taille n on note S la matrice dont le coefficient situé ligne i colonne j est la somme maximale des chemins de $M_{0,0}$ vers $M_{i,j}$, on a donc $S_{0,0} = M_{0,0}$.

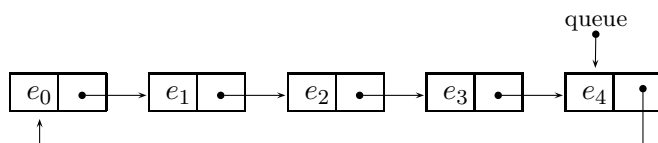
- Q29**– Ecrire la relation de récurrence permettant de calculer les $S_{0,k}$ pour $1 \leq k \leq n - 1$.
- Q30**– Ecrire la relation de récurrence permettant de calculer les $S_{k,0}$ pour $1 \leq k \leq n - 1$.
- Q31**– Montrer que pour i et j non nuls, $S_{i,j} = \max\{S_{i-1,j}, S_{i,j-1}\} + M_{i,j}$.
- Q32**– Ecrire une fonction *réursive* qui calcule de façon descendante les coefficients de la matrice S .
- Q33**– Faire un schéma des appels récursifs de la fonction précédente, que peut-on en conclure ?
- Q34**– Ecrire une fonction OCaml *itérative* `somme_maxi int array array -> int array array` qui prend en argument une matrice M et calcule de façon ascendante les coefficients de la matrice S .

Q35– Déterminer la complexité de cette fonction en fonction n .

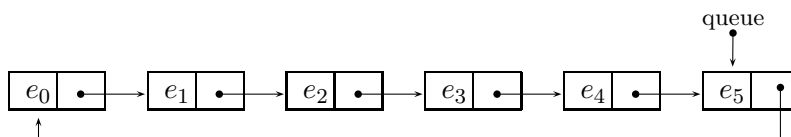
□ **Exercice 4 : Liste chaînée circulaire**

Le langage utilisé dans cette partie est le langage C.

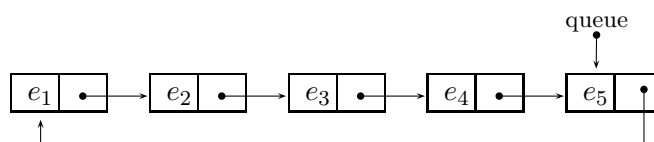
On veut implémenter une structure de données de liste chaînée circulaire avec un pointeur sur la queue telle que représentée ci-dessous :



La queue pointe toujours vers *le dernier élément inséré* ainsi, après l'ajout d'un nouvel élément e_5 , la structure de donnée ci-dessus devient :



Lorsqu'on retire un élément de cette structure de données, on retire le maillon qui suit le dernier élément inséré. Par conséquent, si on retire un élément de la structure de donnée ci-dessus, c'est le maillon contenant e_0 qui est retiré et on obtient :



Afin d'implémenter cette structure de données, on propose d'utiliser les types suivants

```

1 struct maillon
2 {
3     int valeur;
4     struct maillon *suivant;
5 };
6 typedef struct maillon maillon;
7 typedef maillon *liste_circulaire;

```

La liste chaînée circulaire vide est alors représentée par le pointeur NULL.

Q36– En partant d'une liste chaînée circulaire initialement vide, donner les étapes de son évolution après les opérations suivantes :

1. ajouter 12
2. ajouter 6
3. ajouter 7
4. retirer
5. ajouter 42
6. retirer

On précisera la valeur des entiers renvoyés par la fonction **Retirer**.

Q37– Pour ajouter un élément dans une liste circulaire, on propose la fonction ci-dessous :

```

1 void ajouter_errone(liste_circulaire *lc, int val)
2 {
3     maillon *nouveau = malloc(sizeof(maillon));
4     nouveau->valeur = val;
5     nouveau->suivant = (*lc)->suivant;
6     (*lc)->suivant = nouveau;
7     *lc = nouveau;
8 }

```

Indiquer, en justifiant votre réponse, quelle ligne de cette fonction est problématique si la liste circulaire `lc` est vide.

- Q38**– Corriger la fonction `ajouter_errone` afin de traiter convenablement le cas où la liste circulaire `lc` est vide.
- Q39**– Ecrire la fonction `retirer` de signature `int retirer(liste_circulaire *lc)` qui prend en argument une liste circulaire *supposée non vide* et renvoie la valeur du maillon situé après le pointeur de queue en le retirant de la liste circulaire.
- Q40**– Quelle structure de données connue a-t-on implémenté ici ? Justifier et proposer des noms plus appropriés pour les fonctions `ajouter` et `retirer`.
- Q41**– Ecrire une fonction `longueur` de signature `int longueur(liste_circulaire *lc)` qui renvoie le nombre d'éléments d'une liste chaînée circulaire.
- Q42**– Donner, en les justifiant, les complexités des opérations `retirer`, `ajouter` et `longueur`.