Ordre de visite

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve : 3 heures 30 minutes

Juin/Juillet 2023

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Il vous a été donné un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion !). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures !
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple : $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe.*** Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

Les Parties 1 et 2 (OCaml) sont indépendantes des Parties 3 et 4 (C).

Il est précisé au début de chaque partie le langage à utiliser pour l'implémentation. Cette consigne doit impérativement être suivie. Il est à noter que le jury inspectera le code fourni et reproduira l'obtention des résultats à l'aide de votre code.

Ainsi, il est impératif de nous fournir sur votre clé usb votre fichier OCaml, ainsi que l'ensemble de votre code C que l'on compilera avec les commandes indiquées en début de Partie 3.

1 Génération pseudo-aléatoire de structures en OCaml

Vous devez utiliser OCaml tout au long de cette partie.

Étant donné u_0, on définit par récurrence

$$u_{t+1} := 19\ 999\ 991 u_t \pmod{19\ 999\ 999}$$

pour tout $t \in \mathbb{N}$, c'est-à-dire u_{t+1} est le reste de la division euclidienne par 19 999 999 du produit entre 19 999 991 et u_t.

Question 1 *Calculer les valeurs de u_t pour les valeurs de paramètres suivantes :*

a) $u_{50} \pmod{1\ 000}$ ***b)*** $u_{100} \pmod{1\ 000}$

c) $u_{1\ 000} \pmod{1\ 000}$ ***d)*** $u_{5\ 000} \pmod{1\ 000}$

1.1 Graphes orientés

Un graphe orienté fini $G = (V, A)$ est défini par :
- $V \subseteq \mathbb{N}$, son ensemble (fini) de sommets ;
- un ensemble $A \subseteq V^2 \setminus \{(v, v) \mid v \in V\}$ d'arcs consistant en des paires de sommets de V.

On notera que l'on interdit explicitement les boucles, c'est-à-dire des arcs joignant un sommet à lui-même.

Dans un graphe orienté, pour un sommet $x \in V$, on note $N^+(x)$ le voisinage sortant de x, consistant en l'ensemble des sommets atteignables en au plus un arc depuis x :

$$N^+(x) \coloneqq \{y \mid (x, y) \in A\}.$$

De même, le voisinage entrant pour un sommet $x \in V$ sera défini de la façon suivante :

$$N^-(x) \coloneqq \{y \mid (y, x) \in A\}.$$

Par extension, pour un ensemble $U \subseteq V$ de sommets de G, on définira le voisinage sortant de U :

$$N^+(U) \coloneqq \{y \in N^+(x) \mid x \in U\}$$

et entrant de U :

$$N^-(U) \coloneqq \{y \in N^-(x) \mid x \in U\}.$$

1.2 Génération pseudo-aléatoire de graphes orientés

On définit pour tout $n, m \in \mathbb{N}^*$ avec $m < 1\ 000$, le graphe orienté $G(n, m) = (\{0, \ldots, n-1\}, A)$ avec A défini de la façon suivante :

$$A \coloneqq \{(x, y) \mid x \neq y \wedge (u_{n+m+7x+11y} \pmod{1\ 000} < m)\}.$$

Question 2 *Calculer le nombre d'arcs des graphes suivants :*

a) $G(100, 100)$ ***b)*** $G(100, 500)$ ***c)*** $G(1\ 000, 100)$ ***d)*** $G(1\ 000, 500)$

1.3 Ajout de colorations

Étant donné un graphe orienté $G = (V, A)$, une coloration de ce graphe est la donnée d'une fonction $\lambda : V \to \{0, \ldots, p-1\}$. Un graphe coloré est un triplet $C = (V, A, \lambda)$.

À partir d'un graphe $G(n, m)$ défini précédemment, on définit un graphe coloré $C(n, m, p)$ en y ajoutant la fonction λ suivante :

$$v \in V \mapsto u_{5v} \pmod p.$$

La somme de contrôle d'un graphe colorié $C = (V, A, \lambda)$ est définie comme : $|A| + \sum_{v \in V} \lambda(v)$.

Question 3 *Calculer la somme de contrôle des graphes coloriés suivants :*

a) $C(777, 222, 5)$ ***b)*** $C(777, 500, 25)$ ***c)*** $C(1\ 234, 222, 25)$ ***d)*** $C(1\ 234, 500, 75)$

Question à développer pendant l'oral 1 *Décrire la structure de données que vous avez choisie pour représenter les graphes. Votre réponse comportera une estimation de l'espace utilisé en fonction des paramètres n, m et p. Vous devez particulièrement insister sur l'impact du paramètre m dans votre choix.*

2 Observations sur des graphes colorés en OCaml

Vous devez utiliser OCaml tout au long de cette partie.

Étant donné un graphe orienté coloré $C = (V, A, \lambda)$, un chemin de C de longueur k est une suite $(v_0, \ldots, v_k)$ de sommets de V telle que pour tout $0 \leq i < k$, $(v_i, v_{i+1}) \in A$. v_0 est appelé la source de ce chemin.

Une observation sur C de longueur k est une suite $(\lambda(v_0), \ldots, \lambda(v_k))$ de couleurs de C quand $(v_0, \ldots, v_k)$ est un chemin de C de longueur k.

L'ensemble des observations possibles depuis un sommet v du graphe est défini comme :

$$O(v) \coloneqq \{(\lambda(v_0), \ldots, \lambda(v_k)) \mid k \geq 0,\ (v_0, \ldots, v_k) \text{ est un chemin de source } v \text{ (i.e., } v = v_0) \text{ dans } C\}.$$

Une simulation réciproque sur C est une relation binaire R sur V telle que $(v, w) \in R$ si et seulement si :
- $\lambda(v) = \lambda(w)$;
- $\forall v' \in N^+(v),\ \exists w' \in N^+(w),\ (v', w') \in R$;
- $\forall w' \in N^+(w),\ \exists v' \in N^+(v),\ (v', w') \in R$.

Deux sommets v et w sont indistinguables, noté $v \simeq w$, si et seulement si il existe une simulation réciproque R telle que $(v, w) \in R$.

Notre objectif dans cette première partie de l'épreuve est de partitionner les sommets de V suivant le critère d'indistinguabilité.

Question à développer pendant l'oral 2 *Est-ce que $v \simeq w$ est équivalent à $O(v) = O(w)$? Justifier rapidement ou donner un contre-exemple.*

2.1 Raffinement de partition

Soit un ensemble E, une partition $\mathcal{L} = (\mathcal{I}_1, \ldots, \mathcal{I}_l)$ est une liste ordonnée de sous-ensembles disjoints (appelés classes) de E dont l'union est E.

Raffiner une partition $\mathcal{L}$ vis-à-vis d'un ensemble $S \subseteq E$ consiste à diviser chaque classe $\mathcal{I}_\alpha \in \mathcal{L}$ en deux : $\mathcal{I}_\alpha \cap S$ et $\mathcal{I}_\alpha \setminus S$, et ne garder ces ensembles que s'ils sont non vides.

Algorithme 1 Raffinement de partition

Entrée: une partition $\mathcal{L} = (\mathcal{I}_1, \ldots, \mathcal{I}_l)$ d'un ensemble E et un sous-ensemble $S \subseteq E$.
Sortie: une partition raffinée $\mathcal{L}' = (\mathcal{I'}_1, \ldots, \mathcal{I'}_m)$.

```
pour chaque classe I_α faire
    soit J les éléments de I_α qui sont dans S
    retirer J de I_α
    si I_α est vide alors
        remplacer I_α par J
    sinon si J n'est pas vide alors
        ajouter J avant I_α
    fin si
fin pour
renvoyer L
```

On note $\mathsf{R}_E(\mathcal{L}, S)$ l'action de raffiner une partition $\mathcal{L}$ de E par un sous-ensemble S de E à l'aide de l'algorithme 1.

Après avoir raffiné une partition $\mathcal{L}$ par rapport à un ensemble S, plus aucune classe ne chevauche S : pour toute classe $\mathcal{I}_\alpha \in \mathcal{L}'$, $S \cap \mathcal{I}_\alpha = \mathcal{I}_\alpha$ ou $S \cap \mathcal{I}_\alpha = \emptyset$.

Implémentation et test du raffinement de partition. Soit $G(n, m)$ un graphe orienté pseudo-aléatoire et soit $r \geq 0$.
Pour $0 \leq i \leq r$, on note v_i le sommet $u_{n+m+i} \mod n$. Nous allons séquentiellement raffiner V par $N^+(v_i)$, on note $\mathcal{L}_r$ la partition ainsi obtenue :

$$\mathcal{L}_r \coloneqq \mathsf{R}_V(\ldots \mathsf{R}_V(\mathsf{R}_V(V, N^+(v_0)), N^+(v_1)) \ldots, N^+(v_r))$$

Noter que l'on a implicitement identifié V dans la formule précédente avec l'unique partition de V ayant V pour seule classe.

Question 4 *Calculer $|\mathcal{L}_r|$ pour les couples de graphes et de r suivants :*

a) $G(111, 222), 5$ ***b)*** $G(111, 500), 100$ ***c)*** $G(1\ 234, 222), 5$ ***d)*** $G(1\ 234, 500), 100$

2.2 Calcul itératif de la relation $\simeq$

Nous allons maintenant calculer itérativement la relation $\simeq$.

Deux sommets v et w sont indistinguables pour des observations de longueur 0, noté $v \simeq_0 w$, si et seulement si :
— $\lambda(v) = \lambda(w)$.

Deux sommets v et w sont indistinguables pour des observations de longueur au plus $k \geq 1$, noté $v \simeq_k w$, si et seulement si :
— $\lambda(v) = \lambda(w)$;
— $\forall v' \in N^+(v)\ \exists w' \in N^+(w),\ v' \simeq_{k-1} w'$;
— $\forall w' \in N^+(w)\ \exists v' \in N^+(v),\ v' \simeq_{k-1} w'$.

Il est clair que pour tout k, on peut identifier une classe d'équivalence de $\simeq_k$ avec un sous-ensemble de V. On peut alors représenter l'ensemble des classes d'équivalences de la relation $\simeq_k$ par une partition de V.

Nous voulons maintenant nous aider du raffinement de partition pour calculer $\simeq_{k+1}$ à partir de $\simeq_k$.

Question à développer pendant l'oral 3 *Présenter comment s'aider du raffinement de partition pour passer d'une représentation de $\simeq_k$ à une représentation de $\simeq_{k+1}$. Aucune preuve formelle n'est attendue.*

Question 5 *Soit $C(n, m, p)$ un graphe orienté colorié précédemment défini. Calculer le nombre de classes d'équivalence pour $\simeq$ sur :*

a) $C(3\,000, 1, 4)$ ***b)*** $C(3\,001, 2, 4)$ ***c)*** $C(3\,002, 900, 200)$ ***d)*** $C(3\,002, 900, 224)$

Question à développer pendant l'oral 4 *Donner le pseudo-code de l'algorithme que vous avez utilisé pour calculer $\simeq$ sous la forme d'une partition. Justifier de la terminaison de votre algorithme.*

3 Algorithmique des graphes en C

Vous devez utiliser C tout au long de cette partie.

3.1 Code de base fourni et explication du jeu de données

Vous trouverez dans un dossier nommé `base` les fichiers suivants : `makefile`, `graphes.txt`, `main.c`, `graphe.c`, `graphe.h`, `utils.c` et `utils.h`.

Le fichier `makefile` contient les informations à destination de la commande `make` afin de permettre la compilation de votre code. N'y touchez que si vous savez ce que vous faites.

Pour compiler, il suffit d'invoquer la commande `make` depuis le répertoire `base`, par exemple :

```
cd base
make
./main
```

Le fichier `graphes.txt` est en lecture seule, et contient l'ensemble des graphes sur lesquels nous allons travailler (ci-après, appelé le jeu de données). Vous devez donc calculer vos réponses à partir du fichier `graphes.txt`.

Vous n'avez pas besoin de consulter son contenu car les entrées/sorties et le stockage en mémoire des graphes est déjà implémenté. Pour tester votre code et produire le résultat, veillez simplement à calculer vos valeurs à partir du nombre de graphes demandés.

Le fichier `main.c` contient la partie principale du code et c'est dans ce fichier que vous êtes encouragés à écrire vos solutions. Vous pourrez aussi avoir besoin de modifier la structure de votre graphe à partir du fichier `graphe.h`. Enfin, `graphe.c`, `utils.c` et `utils.h` contiennent les primitives de base que nous avons implémentées pour vous.

Les graphes chargés à partir du fichier sont stockés sous forme de liste d'adjacence, plus précisément :

- Chaque structure de graphe `struct graphe` contient un champ `size_t taille`[1] indiquant le nombre de sommets dans ce graphe ; puis un pointeur vers le premier élément d'un tableau de longueur `taille` de `struct sommet`, ordonné par ordre croissant d'identifiants.
  N. B. : Les identifiants sont indicés de 0 à `taille` $-$ 1.
- Chaque `struct sommet` contient des informations relatives au sommet : son identifiant `size_t identifiant` et un pointeur sur le premier élément d'une liste simplement chaînée de `struct arete` représentant sa liste d'adjacence, elle aussi ordonnée suivant les identifiants des sommets ; vous êtes libre d'y ajouter des champs supplémentaires.
- Chaque `struct arete` contient l'identifiant du voisin dans un champ `size_t idvoisin` et un pointeur vers l'élément suivant dans la liste simplement chaînée des voisins.

Les valeurs d'exemple des réponses attendues ont été calculées uniquement à partir des $X = 4$ premiers graphes du jeu de données.

Définitions et suite de l'épreuve. Un graphe non-orienté fini $G = (V, E)$ est défini par :

- $V \subseteq \mathbb{N}$, son ensemble (fini) de sommets ;
- un ensemble $E \subseteq \{e \in 2^V \mid |e| = 2\}$ d'arêtes consistant en des ensembles (de cardinal strictement 2) de sommets de V.

1. Le type de données `size_t` en C est un type d'entiers non signés utilisé entre autres pour stocker la taille de tableaux.

Dans un graphe non-orienté, pour un sommet $x \in V$, on note $N(x)$ le voisinage de x consistant en l'ensemble des sommets reliés à x, c'est-à-dire $N(x) \coloneqq \{y \mid \{x, y\} \in E\}$.

Dans un graphe non-orienté, on appellera un cycle une suite d'arêtes consécutives distinctes (chaîne) dont les deux sommets extrémités sont identiques. Si la chaîne est élémentaire, c'est-à-dire ne passe pas deux fois par un même sommet (les extrémités pouvant néanmoins être identiques), alors on parle de cycle élémentaire. Un graphe est dit acyclique si il ne contient aucun cycle élémentaire.

Tous les graphes du jeu de données sont non-orientés et sans boucles.

3.2 Arbres et forêts

Un arbre est un graphe acyclique connexe, une forêt est un graphe acyclique.

Question 6 *Soit a le nombre d'arbres parmi les graphes du jeu de données. Soit f le nombre de forêts parmi les graphes du jeu de données. Calculer les valeurs suivantes :*

a) a ***b)*** f

Question à développer pendant l'oral 5 *Présenter le pseudo-code de l'algorithme pour les arbres, et faire l'étude de sa complexité temporelle.*

3.3 Ponts

Un pont dans un graphe est une arête dont la suppression entraîne une augmentation du nombre de composantes connexes. C'est à dire qu'il existe deux sommets du graphe qui étaient précédemment joignables, mais qui ne le sont plus quand on supprime cette arête. Notre but est de proposer un algorithme permettant de compter le nombre de ponts dans un graphe.

Pour ce faire, nous allons utiliser une approche basée sur le parcours en profondeur d'un graphe. Nous appellerons "ordre de visite" l'ordre obtenu en numérotant les sommets visités par un parcours en profondeur avec le rang auquel ils ont été visités. L'algorithme 2 présente la construction d'un tel ordre dans un tableau global nommé rang.

Algorithme 2 Parcours en profondeur (DFSRANG) – Construction du rang

Entrée: un graphe non-orienté $G = (V, E)$, un sommet r et un entier i.
Sortie: le rang du dernier sommet visité à partir de r par le parcours en profondeur.

```
rang[r] ← i
marquer r comme visité
pour chaque voisin v de r faire
    si v n'a pas déjà été visité alors
        i ← DFSRANG(G, v, i + 1)
    fin si
fin pour
renvoyer i
```

Pour identifier les ponts dans un graphe, nous allons nous baser sur l'observation suivante. Disons que l'on est au cours de l'exécution d'un parcours en profondeur sur un graphe donné, et que l'on est en train d'itérer sur les voisins d'un sommet r. L'arête actuelle (r, v) est un pont si et seulement si, en excluant cette arête, ni le sommet v, ni les sommets visités à partir de v par le parcours en profondeur n'ont d'arêtes vers le sommet r ou un sommet visité avant r. En effet, cette condition signifie qu'il n'existe pas d'autre possibilité que l'arête (r, v) pour aller de r à v.

Au sein d'une même composante connexe, être visité avant r signifie avoir un rang plus petit que celui de r, et être visité à partir de v signifie avoir un rang compris entre $\text{rang}(v) + 1$ et le rang retourné par le parcours en profondeur sur v.

Question à développer pendant l'oral 6 *Présenter le pseudo-code l'algorithme en découlant, et faire l'étude de sa complexité temporelle.*

Question 7 *On note p le nombre total de ponts pour l'ensemble des graphes du jeu de données. Implémenter la méthode ci-dessus afin de calculer la valeur suivante :*

a) p

3.4 Points d'articulation

Un point d'articulation est un sommet dont la suppression entraîne une augmentation du nombre de composantes connexes. C'est-à-dire qu'il existe deux sommets du graphe qui étaient précédemment joignables, mais qui ne le sont plus quand on supprime ce point d'articulation. Notre but est de proposer un algorithme permettant de compter le nombre de points d'articulation dans un graphe.

Nous allons à nouveau utiliser une approche basée sur le parcours en profondeur d'un graphe. Nous appellerons l'arbre d'un parcours en profondeur l'arbre dans lequel chaque sommet visité par un parcours en profondeur a pour parent le sommet depuis lequel il a été visité. L'algorithme 3 présente la construction d'un tel arbre, représenté dans un tableau global nommé parent, associant à chaque sommet l'identifiant de son parent.

Algorithme 3 Parcours en profondeur (DFSPARENTS) – Construction de l'arbre

Entrée: un graphe non-orienté $G = (V, E)$, un sommet r et le parent p.
Sortie: le sous-arbre du parcours en largeur, enraciné à r, représenté dans parent.

```
parent[r] ← p
marquer r comme visité
pour chaque voisin v de r faire
    si v n'a pas déjà été visité alors
        DFSPARENTS(G, v, r)
    fin si
fin pour
```

Pour identifier les points d'articulation dans un graphe, nous allons nous baser sur l'observation suivante. Dans l'arbre du parcours en profondeur, un sommet r est un point d'articulation si et seulement si l'une des deux conditions suivantes est vraie :

- r est la racine de l'arbre du parcours en profondeur et il a au moins deux fils dans cet arbre ;
- r n'est pas la racine de l'arbre du parcours en profondeur et il a un fils v tel qu'aucun sommet dans le sous-arbre du parcours en profondeur enraciné à v n'ait une arête vers un ancêtre de r.

Question à développer pendant l'oral 7 *Présenter le pseudo-code l'algorithme en découlant, et faire l'étude de sa complexité temporelle.*

Question 8 *On définit a le nombre de points d'articulation pour l'ensemble des graphes du jeu de données. Implémenter la méthode ci-dessus afin de calculer la valeur suivante :*

a) a

3.5 Plus longs chemins

La projection dirigée $D_G^< = (V, E_G^<)$ d'un graphe non-orienté $G = (V, E)$, par rapport à une relation d'ordre totale et stricte $<$ sur V, est définie par :

- la donnée V de son ensemble de sommets ;
- d'un ensemble $E_G^< \coloneqq \{(u, v) \in E^2 \mid \{u, v\} \in E, u < v\}$ d'arcs consistant en des paires de sommets de V.

On considère maintenant la projection dirigée des graphes du jeu de données, par rapport à la relation ordonnant les sommets par identifiants croissants.

Question 9 *Pour chaque graphe du jeu de données, on calcule la longueur maximale d'un chemin. Puis on somme toutes ces valeurs pour obtenir l. Calculer la valeur suivante :*

a) l

Vous serez évalué sur votre capacité à proposer, et implémenter, une solution la plus efficace possible.

Question à développer pendant l'oral 8 *Présenter le pseudo-code de votre algorithme pour calculer la longueur maximale d'un chemin dans un des graphes ainsi obtenus, donner sa complexité temporelle et prouver sa correction. Expliciter comment vous avez adapté votre algorithme au vu de la structure de données utilisée dans le code de base fourni.*

4 Détection de graphes cordaux en C

Vous devez utiliser C tout au long de cette partie.

Soit $G = (V, E)$ un graphe non-orienté et soit $V' \subseteq V$ un sous-ensemble de V. On définit le sous-graphe induit par V' sur G, dénoté $G(V')$, comme étant le graphe $(V', E \cap 2^{V'})$, c'est-à-dire G restreint aux sommets de V' et à l'ensemble des arêtes de E reliant les sommets de V' entre eux.

Une clique dans G est un sous-ensemble de sommets de V deux à deux reliés entre eux par E.

Un graphe est dit cordal quand pour tous ses cycles élémentaires de quatre sommets ou plus, il existe deux sommets non consécutifs du cycle reliés par une arête. Une définition équivalente consiste à s'assurer de l'existence d'un ordre d'élimination simplicial, c'est-à-dire un ordre total $\sigma = (v_1, \ldots, v_n)$ sur les sommets de G tel que pour tout $v_i \in V$, le voisinage de v_i dans $G(v_{i+1}, \ldots, v_n)$ – le sous-graphe induit par $\{v_{i+1}, \ldots, v_n\}$ sur G – est une clique. On admettra qu'un graphe est cordal si et seulement si il admet un ordre d'élimination simplicial.

Question à développer pendant l'oral 9 *Présenter succinctement la méthode que vous avez utilisée pour construire les permutations d'une liste d'entiers, et prouver sa correction.*

Question à développer pendant l'oral 10 *Étant donnée une permutation des sommets de V présentée sous la forme d'une liste d'entiers, et compte tenu de la représentation du graphe utilisée dans ce sujet :*

- *Quelle est la complexité en temps et en espace pour vérifier que cette permutation est bien un ordre simplicial ?*
- *Peut-on obtenir une meilleure complexité temporelle ? Quelle serait alors la complexité en espace ? Il ne vous est pas demandé d'implémenter cette solution alternative.*

Question 10 *Soit c le nombre de graphes cordaux parmi les $Y_0 = 10$ premiers graphes du jeu de données. Implémenter cette caractérisation afin de calculer la valeur suivante :*

a) c

Fiche réponse type : Ordre de visite

$\widetilde{u_0}$: 5

Question 1

a) 640

b) 757

c) 105

d) 504

Question 2

a) 1015

b) 5048

c) 102573

d) 501947

Question 3

a) 134801

b) 313214

c) 353383

d) 801935

Question 4

a) 32

b) 111

c) 55

d) 1234

Question 5

a) 2802

b) 2947

c) 200

d) 3002

Question 6

a) 1

b) 2

Question 7

a) 9

Question 8

a) 5

Question 9

a) 13

Question 10

a) 2

Fiche réponse : Ordre de visite

Nom, prénom, u_0 : .....

Question 1

a)

b)

c)

d)

Question 2

a)

b)

c)

d)

Question 3

a)

b)

c)

d)

Question 4

a)

b)

c)

d)

Question 5

a)

b)

c)

d)

Question 6

a)

b)

Question 7

a)

Question 8

a)

Question 9

a)

Question 10

a)