MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

EAE SIN 3

SESSION 2022

AGRÉGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

CONCEPTION PRÉLIMINAIRE D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice autorisée selon les modalités de la circulaire du 17 juin 2021 publiée au BOEN du 29 juillet 2021.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	1417A	103	1268

Ce sujet est composé de 32 pages :

- d'un dossier questionnement de la page 1 à la page 16 ;
  - partie 1 de la page 3 à la page 8 ;
  - partie 2 de la page 9 à la page 11 ;
  - partie 3 de la page 12 à la page 16 ;

- d'un dossier technique (DT) de la page 17 à la page 26 ;

- d'un dossier réponse (DR) de la page 27 à la page 32.

Conseils aux candidats :

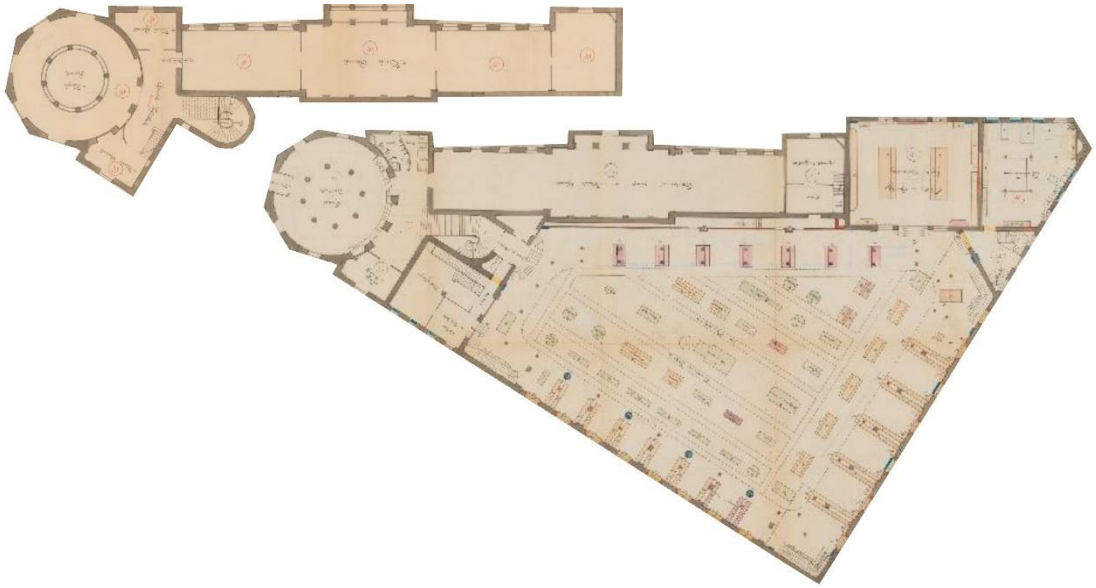
- les trois parties du questionnement sont indépendantes ;
- un parcours attentif de l'ensemble du document est conseillé avant de composer ;
- la présentation des programmes doit respecter les mots clés du langage cible ainsi que l'indentation des structures algorithmiques ;
- les réponses doivent être présentées avec clarté, rigueur et concision.

Tournez la page S.V.P.

VISITES ENRICHIES D'UN MUSEE

Depuis plusieurs années, les musées cherchent à mieux connaître leurs visiteurs afin de leur proposer des contenus adaptés à leurs attentes. Certains visiteurs planifient des visites thématiques, d'autres recherchent uniquement les œuvres majeures, les nouveautés ou leurs œuvres préférées dans un circuit court.

L'objectif pour le musée est double : les offres peuvent être adaptées plus finement aux différents publics et des contenus additionnels ciblés peuvent être proposés à la fin de la visite.



Plan du musée Guimet (Lyon), Tony Blein architecte, 19 mars 1912
domaine public, archives municipales de Lyon

Cette collecte d'informations doit être transparente et simple du point de vue du visiteur. L'application statique pour smartphone proposée depuis plusieurs années est récemment mise à jour dans le but de rendre son contenu plus dynamique et éventuellement personnalisable. Son installation et son utilisation sont étudiées pour un usage « grand-public ».

Les méthodes, notations et figures employées dans ce sujet (questionnement, documents techniques et documents réponses) sont inspirées de l'article de recherche Kontarinis, A., Zeitouni, K., Marinica, C. et al. Towards a semantic indoor trajectory model: application to museum visits. Geoinformatica 25, 311–352 (2021).

https://doi.org/10.1007/s10707-020-00430-x

PARTIE 1. LOCALISATION DES VISITEURS

La localisation des visiteurs est assurée en croisant les informations issues de plusieurs sources. Les utilisateurs de l'application sur Smartphone utilisent une application capable de les localiser. Le musée est aussi équipé d'un système de vidéosurveillance couplé à un logiciel d'analyse d'images.

Les objectifs de cette vidéosurveillance partiellement automatisée sont doubles :

- les cohortes de visiteurs, les attroupements et les déplacements majoritaires peuvent être facilement identifiés,
- les algorithmes d'analyse d'image peuvent mettre en évidence les comportements à risques envers les œuvres d'art avec pour but de diriger rapidement la surveillance humaine.

Q1. Ces observations automatisées ont fait l'objet d'une déclaration à la CNIL. Préciser en quelques lignes le cadre règlementaire autour de ces collectes anonymes d'informations et autour de la création d'une expérience personnalisée.

Sous-partie 1.1. Observation des bornes Wi-Fi à proximité

Le réseau Wi-Fi du musée est constitué de 56 bornes réparties sur les différentes salles. Lors de la conception de cette couverture Wi-Fi, deux stratégies différentes ont été étudiées : chaque borne dispose de son propre SSID ou toutes les bornes diffusent le même SSID :

- l'affectation d'un SSID propre à chaque point d'accès (par exemple *MUSEE-001*, *MUSEE-002*…) est la plus simple à réaliser, le visiteur associe son terminal à chaque borne ;
- la diffusion d'un SSID unique (par exemple *MUSEE*) sur tout l'espace du musée a pour conséquence la réalisation d'une association unique avec le réseau, l'association est automatiquement renouvellement par le terminal du visiteur pour chaque borne rencontrée.

L'application du musée peut obtenir la liste des point d'accès à proximité de l'appareil mobile en sollicitant un scan auprès du système d'exploitation. La documentation de la classe *ScanResult* est disponible dans le document technique DT1. Dans la plupart des systèmes d'exploitation, l'obtention de ces informations requiert une permission ou un privilège spécifique pour l'application.

Q2. Justifier le choix d'un SSID unique pour tout le musée. Préciser les informations utiles permettant à l'application d'identifier individuellement les points d'accès à proximité.

Q3. Définir toutes les informations nécessaires à l'application lui permettant de calculer par triangularisation la position du terminal du visiteur. Justifier l'autorisation préalable à l'obtention des informations. Critiquer la précision obtenue dans le bâtiment.

L'association entre le terminal et la borne Wi-Fi est un processus standardisé dans la norme IEEE 802.11. Une explication des échanges effectués est présente dans le document technique DT2.

Tournez la page S.V.P.

Q4. D'après cette description, le processus de connexion fait apparaître trois états. Tracer la machine à états correspondante pour le terminal détenu par le visiteur. Indiquer pour chaque état les actions associées et pour chaque transition la condition associée.

Ces échanges entre le terminal et le point d'accès sont prévus pour être sécurisés.

Q5. Lister les protocoles d'authentifications cités dans ce document technique. Indiquer les principales étapes faisant suite à ces six échanges pour obtenir une connexion pleinement sécurisée. Nommer l'algorithme de chiffrement utilisé par le Wi-Fi.

Q6. Discuter la faisabilité de la localisation d'un terminal du point de vue du réseau en se basant sur les requêtes émises par le terminal pour découvrir les points d'accès du réseau.

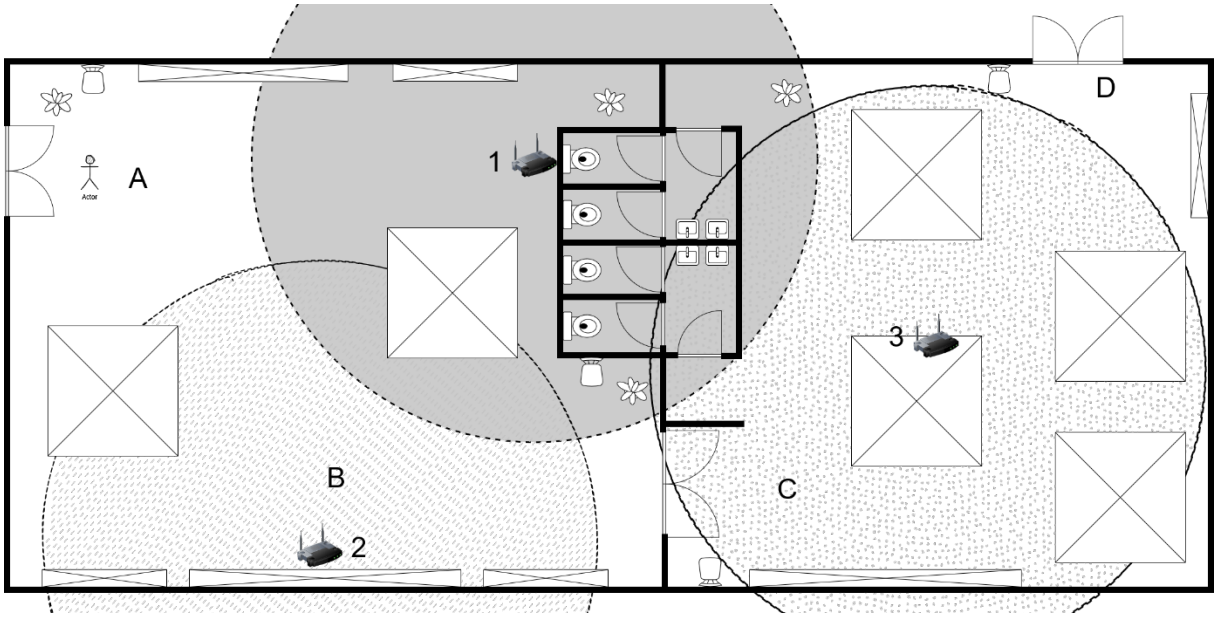
Une fois associé à un point d'accès et dans le but de limiter la consommation d'énergie du terminal du visiteur, sa localisation ne peut plus être effectuée par des scans répétitifs des points d'accès à proximité.

La stratégie de localisation évolue de la manière suivante : une fois associés, le point d'accès et le terminal sont connectés et peuvent échanger des informations. Le système d'exploitation du terminal relève pour chaque trame reçue le niveau de réception RSSI.

Q7. Compléter le code du document réponse DR1 permettant d'obtenir le niveau de réception du point d'accès associé.

Q8. Dans le document réponse DR1, préciser l'opération réalisée par la seconde ligne de code « WifiManager wifiManager = (WifiManager) service; ». Tracer le diagramme de classe correspondant.

La figure ci-dessous représente deux salles du musée couvertes par trois points d'accès Wi-Fi (numérotés de 1 à 3). Les cercles centrés sur chacune des trois bornes représentent la zone où le niveau de réception (RSSI) à -50 dBm est garanti. Ce niveau de puissance reçue permet une « bonne » qualité dans les communications et le terminal reste associé au point d'accès courant. Lorsque le niveau de réception est inférieur à -50 dBm, le système d'exploitation du terminal du visiteur scanne à nouveau tous les réseaux à proximité.



Le visiteur suit une trajectoire passant par les points A, B, C et D présents sur cette figure.

Q9. Estimer la précision de la localisation du visiteur pour ces quatre points définis en se basant sur les trois bornes présentes.

Sous-partie 1.2. Analyse automatisée de la vidéosurveillance

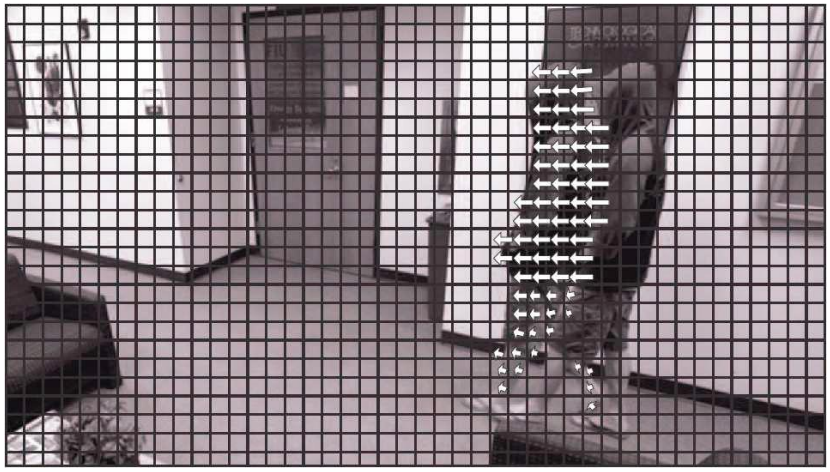
Le réseau de télésurveillance du musée filme les œuvres pour aider à la détection des comportements à risques de la part des visiteurs. Cette détection est effectuée pour partie automatiquement dans le but d'attirer l'attention de surveillants humains.

Toutes les caméras sont reliées à un centre informatisé exécutant plusieurs traitements sur les flux vidéos. La résolution des images est insuffisante pour identifier les individus. Ces flux vidéos permettent de suivre les déplacements des visiteurs. Un déplacement est modélisé comme étant la succession de plusieurs positions d'un même individu au cours du temps. Extraire la position des visiteurs d'une image unique est une tâche relativement aisée. La principale difficulté réside dans la séparation des mouvements de plusieurs individus et l'extraction de plusieurs mouvements distincts dans une suite d'image.

Les nombreux flux vidéos sont encodés par les caméras à l'aide d'un codec vidéo dont l'algorithme de compression est basé sur le déplacement apparent de groupes de pixels entre plusieurs images consécutives. Ces mouvements de groupes de pixels sont appelés *motion vectors*.

La figure ci-dessous superpose deux images d'un même individu, le quadrillage met en évidence les groupes de pixels considérés par l'algorithme de compression vidéo, les flèches blanches représentent les *motion vectors*.

Tournez la page S.V.P.



Source : Chen, Ruei-Xi & Zhao, Wei & Fan, Jeffrey & Davari, A.. (2009). Vector Bank Based Multimedia Codec System-on-a-Chip (SoC) Design. 515-520. 10.1109/I-SPAN.2009.74.

Afin de limiter la puissance informatique nécessaire aux traitements des nombreux flux vidéos, ceux-ci ne sont pas décompressés. Les informations de déplacement sont directement prélevées du flux brut pour être retraitées.

Le point d'entrée de l'algorithme est une matrice de 128 colonnes sur 96 lignes. Chaque élément de la matrice est un 2-uplet formé de deux nombres caractérisant respectivement les déplacements horizontaux et verticaux de ce groupe de pixels entre deux images successives.

Q10. Proposer un type de données adapté à la représentation des déplacements horizontaux et verticaux stockés pour chaque élément de la matrice.

Dans le but de simplifier le stockage en mémoire de cette matrice, celle-ci est linéarisée sous la forme d'un tableau. La correspondance entre les indices *i* des éléments du tableau et les coordonnées (ligne *l*, colonne *c*) de l'élément dans la matrice est la suivante :

	***c* = 0**	***c* = 1**	**…**	***c* = 126**	***c* = 127**
***l* = 0**	i = 0	*i* = 96	…	*i* = 12096	*i* = 12192
***l* = 1**	*i* = 1	*i* = 97	…	*i* = 12097	*i* = 12193
…	…	…	…	…	…
***l* = 94**	*i* = 94	*i* = 190	…	*i* = 12190	*i* = 12286
***l* = 95**	*i* = 95	*i* = 191	…	*i* = 12191	*i* = 12287

Q11. En déduire l'occupation mémoire de cette matrice. Définir les fonctions de conversions *f* et *g* telles que $i = f(c,l)$ et $(c,l) = g(i)$.

Dans un premier temps, un visiteur vu par la caméra est supposé indéformable et en translation uniforme. Tous les groupes de pixels formant ce visiteur à l'image sont adjacents et caractérisés par les mêmes *motion vectors* (valeurs identiques pour les translations horizontales et verticales).

Q12. Dans le document réponse DR2, compléter le pseudocode de la fonction « coloriage ». Celle-ci associe une même couleur aux groupes adjacents ayant les mêmes déplacements.

Q13. Dans le document réponse DR3, compléter le pseudocode parcourant toute la matrice de déplacement à la recherche d'objets non coloriés

Les visiteurs ont en moyenne une hauteur apparente de 15 éléments et une largeur apparente de 5 éléments dans la matrice. Le parcours de l'image s'effectuant principalement de haut en bas puis de gauche à droite, le premier élément détecté pour un objet est probablement situé vers le haut de cet objet.

Q14. Evaluer la profondeur maximale des appels récursifs pour ce cas moyen. Evaluer la complexité de la fonction « coloriage » puis de l'algorithme vis-à-vis de la taille apparente de l'objet et du nombre d'objets.

Q15. Critiquer la qualité des résultats issus de cet algorithme par rapport à l'allure des *motion vectors* extraits de l'image en début de cette sous-partie. En déduire le nombre d'objets effectivement détectés par cette technique par rapport au nombre réel de visiteurs. Proposer des pistes d'améliorations pour limiter les défauts de cet algorithme.

Une image est normalement constituée de moins de 256 objets détectés. Une fois chaque objet en mouvement colorié, il est possible d'estimer, pour chaque objet :

- sa validité, une surface apparente inférieure à 30 éléments est considérée comme seuil de rejet de cet objet,
- son centre comme étant la moyenne des coordonnées (x,y) des éléments ayant la couleur de cet objet.

Q16. Dans le document réponse DR3, compléter le pseudocode d'estimation de la validité et de la position moyenne de chaque objet.

Le sol de la salle est supposé plan et horizontal. Les coordonnées, orientation et angle d'ouverture de la caméra sont connus. Toutes les caméras sont fixées au plafond et orientées dans une direction plongeante : leur orientation leur permet de filmer le sol et les murs.

La position réelle du centre du visiteur est supposée être à une hauteur de 85 cm par rapport au sol. L'objectif est de déterminer la position réelle du visiteur (x,y,z) dans la salle à partir de la position vue par la caméra du centre de l'objet.

L'objectif de la caméra est supposé linéaire : l'angle apparent de l'objet vu par la caméra est transformé linéairement en pixel. L'image vue par la caméra peut donc être aisément transformé vers un repère polaire.

Q17. Représenter la scène (salle, visiteur et caméra) par un schéma approprié. Justifier l'existence d'une position réelle pour certaines positions vues par la caméra. Préciser les hypothèses et les limites de votre modèle.

Q18. Discuter de la linéarité de ce modèle. Proposer un protocole expérimental permettant d'étalonner les positions réelles des visiteurs en fonction des positions apparentes vues par la caméra.

L'angle de vue de chaque caméra est limité et ne permet d'obtenir une trajectoire sur un temps suffisamment long. Au cours de son déplacement un visiteur est filmé successivement par plusieurs caméras.

Q19. Proposer des améliorations à l'algorithme du document réponse DR3 lui permettant d'annoter les visiteurs détectés

Sous-partie 1.3. Recoupement des observations

Q20. Discuter de la faisabilité technique et réglementaire du regroupement des informations entre les deux sources décrites dans cette partie

PARTIE 2. STOCKAGE DES INFORMATIONS

Les positions successives des visiteurs sont décrites selon la structure hiérarchique représentée dans le document technique DT3 :

- le musée est constitué d'un ou plusieurs complexes, nommés par un chiffre,
- chaque complexe est constitué d'un ou plusieurs bâtiments, nommés par une lettre,
- chaque bâtiment est constitué d'un ou plusieurs étages, nommés par un chiffre,
- chaque étage est constitué d'une ou plusieurs salles, nommées par une lettre,
- chaque salle est constituée d'une ou plusieurs zones d'intérêt (*RoI = Region of Interest*), nommés par un chiffre.

Q21. Traduire cette représentation structurée en un diagramme de classes. Préciser la nature des liens présents entre les différentes entités du diagramme.

Ces positions sont stockées dans une base de données de type *SQL* (*MariaDB*). Les contraintes d'intégrités doivent être gérées par la base de données et donc décrites dans son schéma et présentes lors des requêtes structurelles.

Les lettres et chiffres attribués sont uniques dans la zone considérée : il n'y a qu'une salle « E » dans l'étage « 2 » du bâtiment « A » du complexe « 1 ». Cependant, cette unicité n'est pas garantie entre plusieurs zones : par exemple, il existe plusieurs étages « 2 » dans différents bâtiments.

Pour la suite, chaque complexe/bâtiment/étage/salle/zone est identifié par un identifiant unique, classiquement obtenu via un index auto-incrémenté nommé « identifiant » dans chaque table. Les niveaux inférieurs (respectivement bâtiment, étage, salle ou zone) font référence à l'identifiant nommé « parent » de leur niveau supérieur (respectivement complexe, bâtiment, étage ou salle).

Q22. Affiner le diagramme précédent pour faire apparaître les attributs « identifiant » et « parent », utiliser les liens appropriés entre les entités.

La structure du musée n'est pas fixe à l'échelle de plusieurs années : de nouveaux bâtiments peuvent être construits, des salles peuvent être fermées pour rénovation, la disposition des salles dans un étage peut être changée en profondeur grâce à des cloisons mobiles…

La cohérence de la base de données doit être assurée à tout instant : les zones d'intérêt d'une salle fermée n'existent plus et doivent être supprimées, le déplacement de plusieurs œuvres (formant une zone d'intérêt) dans une autre salle peut être réalisé en conservant la zone actuelle.

Pour chaque table :

- la colonne « identifiant » est un entier auto-incrémenté et est la clé primaire,
- la colonne « parent » est un entier et est une clé étrangère faisant référence à la colonne « identifiant » de la table définissant le « parent ».

Le document technique DT4 rappelle les éléments de syntaxe SQL pour créer une table.

Q23. Ecrire le code SQL pour créer les tables « complexe », « batiment », « etage », « salle », « zone », nécessaires aux définitions des différentes localisations.

Chaque localisation d'un visiteur est une ligne dans la table « localisation ». Cette table est déjà créée et est constituée des colonnes définies de la façon suivante :

- « identifiant », un identifiant unique pour cette localisation, c'est un entier auto-incrémenté.
- « visiteur », une référence vers l'identifiant unique du visiteur (anonymisé) , défini comme un entier dans la colonne « identifiant » de la table « visiteur »,
- « instant », un instant d'observation (TIMESTAMP) sous la forme d'une date et d'une heure (précis à la seconde près), ajouté automatiquement pour chaque insertion,
- « zone », une référence vers l'identifiant unique de la zone d'intérêt, défini comme un entier dans la colonne « identifiant » de la table « zone ».

Le 09 mars 2022 à 12h35m36s (maintenant), le visiteur 1368356 a été détecté dans la zone 1745.

Q24. Ecrire la requête SQL correspondante pour insérer ce nouvel enregistrement dans la table « localisation ». Déterminer la requête retournant la liste de tous les identifiants de visiteurs présents entre le 10 septembre 2021 11h00 et le 10 septembre 2021 12h00 dans l'étage d'identifiant 10.

Les requêtes filtrant une partie du musée font apparaître de nombreuses opérations répétitives. En utilisant les vues, il est possible de créer une table virtuelle basée sur la table « localisation » décrite ci-dessus et faisant apparaître les colonnes virtuelles « complexe », « batiment », « etage » et « salle ». Le document technique DT5 expose les principaux éléments de syntaxe nécessaire à la création d'une vue.

Cette vue est basée sur une requête préenregistrée. Comme tout élément préenregistré, il est possible d'exécuter cette requête comme l'utilisateur l'ayant créé ou comme l'utilisateur accédant à la base de donnée.

Q25. A l'aide des éléments de syntaxe donnés, extraire les deux paramétrages possibles. Préciser les avantages, les inconvénients et éventuels risques pour chaque paramétrage.

L'objectif est de créer une vue « details_localisation » exposant la localisation complète de chaque localisation (complexe, bâtiment, étage, salle, zone). Les colonnes de cette vue sont les suivantes :

identifiant	instant	visiteur	zone	salle	etage	batiment	complexe

Q26. Déterminer la requête permettant de créer la vue « details_localisation » et ses huit colonnes. En utilisant cette vue comme une table classique, déterminer la requête retournant la liste de tous les identifiants de visiteurs présents entre le 10 septembre 2021 11h00 et le 10 septembre 2021 12h00 dans l'étage d'identifiant 10.

Les positions des visiteurs sont enregistrées toutes les dix secondes dans la table « localisation ». Le musée peut accueillir plusieurs millions de visiteurs à l'année, la plupart étant présent pendant cinq heures dans l'établissement.

L'algorithme d'évaluation des trajectoires des visiteurs étudié dans la partie suivante nécessite une présentation plus concise des informations. Les localisations successives d'un même visiteur dans une même zone d'intérêt doivent être regroupées sous la forme d'une table « resume_localisation » ayant les colonnes suivantes :

instant_entree	instant_sortie	visiteur	zone	salle	etage	batiment	complexe

L'instant d'entrée est défini comme le plus petit instant de localisation dans cette zone, l'instant de sortie comme le plus grand instant de localisation dans cette zone.

Q27. Créer la vue « resume_localisation » en créant une requête de sélection capable de grouper les lignes similaires de la table « localisation » et d'extraire les instants d'entrée et de sortie.

Q28. Evaluer le nombre d'enregistrements à l'année, proposer des améliorations à la structure pour favoriser la vitesse d'exécution des requêtes de sélection vues dans cette partie.

Tournez la page S.V.P.

PARTIE 3. ANALYSE DES TRAJECTOIRES

Les trajectoires des visiteurs sont décrites dans un premier temps comme des traces. Ce sont des listes de présences datées dans des zones définies du musée :

$$presence = (Z, S, E, B, C, tstart, tend, A)$$

$$trace = \left(Z_j, S_j, E_j, B_j, C_j, tstart_j, tend_j, A_j\right)_{j\in[0,n-1]}$$

Avec :

- Z est l'identifiant d'une zone d'intérêt du musée (c'est un entier « int »),
- S est l'identifiant d'une salle du musée (c'est un entier « int »),
- E est l'identifiant d'un étage du musée (c'est un entier « int »),
- B est l'identifiant d'un bâtiment du musée (c'est un entier « int »),
- C est l'identifiant d'un complexe du musée (c'est un entier « int »),
- $tstart$ est l'instant d'entrée d'un visiteur dans cette zone,
- $tend$ est l'instant de sortie d'un visiteur de cette zone,
- A est une liste dynamique d'annotation de la présence de ce visiteur dans cette zone.

Pour faciliter les traitements de cette partie, les données d'une journée sont entièrement chargées en mémoire. Certains traitements étant exigeants, un langage de programmation ayant une exécution rapide est privilégié, tel que C++. Pour assurer la sûreté de fonctionnement, la gestion automatique de la mémoire est requise et l'utilisation de pointeurs est à proscrire.

Le document technique DT6 contient des exemples d'utilisation des flux et des instants, représentés avec une résolution de la seconde. Le document technique DT7 expose les codes habituels d'utilisation des tableaux en C++20.

Q29. Valider que le type « seconds » permet de décrire tous les instants et tous les intervalles d'une journée à l'échelle de la seconde. Expliquer la différence entre les tableaux « std::array » et les tableaux « std::vector ».

Q30. Dans le document réponse DR4, déclarer les attributs de la structure « presence » et le type « trace ».

Q31. Ce programme est compilé pour être exécuté sur l'architecture matérielle x86_64. Estimer l'empreinte mémoire des types « presence » et « trace » lorsque les listes dynamiques sont vides. Préciser vos hypothèses.

Sous-partie 3.1. Cohérence des traces

Les traces sont stockées en mémoire en respectant la causalité : un parcours par ordre croissant des éléments d'une trace induit un ordre croissant dans les instants d'entrée.

La première validation de la cohérence des traces consiste à s'assurer que l'instant de sortie d'une zone est toujours inférieur à l'instant d'entrée dans la zone suivante.

Q32. Expliquer le rôle de l'opérateur « & » utilisé dans la déclaration du paramètre « t » de la fonction « corrige_trace » (dans le document réponse DR4). Justifier son emploi.

Q33. Dans le document réponse DR4, compléter le code de la fonction « corrige_trace ».

Certaines traces sont incomplètes : le visiteur peut avoir éteint l'application du musée et/ou ses déplacements peuvent ponctuellement ne pas avoir été reconnus par certaines caméras. Néanmoins les trajectoires manquantes peuvent être complétées car le cheminement dans le musée est majoritairement fléché : les visiteurs passent majoritairement dans les différentes zones dans un circuit établi.

Cette interpolation de la trace ne permet pas de connaître précisément le temps passé dans chacune des zones. Le temps dans chacune de ces zones interpolées est calculé comme étant le temps d'absence divisé par le nombre de zones interpolées.

L'application met à disposition une fonction « interpole_zone » qui prend en paramètres les identifiants des zones dans deux présences consécutives et qui retourne le vecteur d'identifiants des zones à insérer entre ces présences. Si les deux zones passées en paramètres sont adjacentes, le vecteur de retour est vide « size() == 0 ».

La fonction « details_zone » permet de récupérer les identifiants de complexe, bâtiment, étage et salle pour une zone donnée.

Q34. Dans le document réponse DR5, compléter le code de la fonction « interpole_trace »

Une fois corrigées et interpolées, les traces des visiteurs sont maintenant considérées comme des trajectoires desquelles des données peuvent être extraites.

Sous-partie 3.2. Exploitation directe des données

Dans un premier temps le musée cherche à extraire des indicateurs quantitatifs de ces trajectoires sans les altérer :

- nombre de visiteurs pour les différentes parties du musée,
- temps moyen pour les différentes parties,
- temps moyen par visiteur,
- fréquentation par tranche horaire pour les salles du musée…

Q35. Ecrire en C++ la fonction « temps_moyen_visiteur » capable de calculer le temps moyen de visite du musée, précis à la seconde, par visiteur pour toutes les trajectoires d'une journée (vecteur de « trace »).

Q36. Les données étant chargées en mémoire, le calcul de ces indicateurs sont des traitements assez longs. Proposer et justifier une organisation permettant une réduction sensible du temps d'exécution apparent.

Les deux premiers indicateurs peuvent être calculés conjointement : le temps moyen est le rapport entre le temps cumulé et le nombre de visiteurs.

Une approche naïve consiste en l'accumulation pour chaque complexe, chaque bâtiment, chaque étage, chaque salle et chaque zone du temps cumulé de visite et du nombre de visiteurs. Ce cumul nécessite de parcourir toutes les traces de tous les visiteurs d'une journée.

Q37. Evaluer la linéarité/complexité du temps d'exécution de cet algorithme naïf en fonction du nombre de complexes, bâtiments, étages, salles, zones et visiteurs. Conclure sur la faisabilité de cette approche.

Le document technique DT8 propose un algorithme de calcul du nombre de visiteurs et du temps moyen par étage.

Au cours de la journée considérée, le musée a été fréquenté par 10 000 visiteurs. Lors de l'exécution de cet algorithme, la fréquentation d'un étage est calculée à 4 323 562 visiteurs. Les identifiants des étages et les trajectoires des visiteurs sont correctes.

Q38. Identifier et expliquer textuellement le dysfonctionnement de cet algorithme. La correction de cet algorithme n'est pas requise.

Lors de l'exécution de la fonction « occupation_etage », un doublement approximatif de la mémoire utilisée par le processus a été diagnostiqué.

Le code d'appel est le suivant :

```
int main()
{
      vector<trace> traj = charger_trajectoire();
      vector<seconds> t;
      vector<int> v;
      occupation_etage(traj, t, v);
      …
      return 0;
}
```

Q39. Identifier et expliquer la cause de ce dysfonctionnement. Recopier et adapter les quelques lignes de code nécessaires à sa résolution.

Sous-partie 3.3 Exploration des données et prospective

Les trajectoires obtenues peuvent également être recoupées afin de mettre en évidence des points communs et dégager spontanément les profils des visiteurs. Cette création d'information est basée sur une exploration des données existantes (*data-mining*) dans le but de créer de nouvelles informations.

La trace d'un visiteur est définie selon les variables suivantes :

$$trace = \left(Z_j, S_j, E_j, B_j, C_j, tstart_j, tend_j, A_j\right)_{j\in[0,n-1]}$$

L'empreinte temporelle d'une trace est définie comme le temps écoulé $tdelta_i$ dans les différentes zones visitées z_j. Par définition, un temps nul est affecté à une zone non-visitée :

$$empreinte_{trace} = \left(tdelta_i = tend_j - tstart_j\right)_{\substack{i=z_j \\ i\in[1,nb_{zones}]}}$$

Q40. Déterminer la relation de corrélation de l'empreinte de deux traces (aucun code requis). Préciser l'unité du résultat. Evaluer le nombres de calculs nécessaires pour calculer la corrélation entre tous les visiteurs deux-à-deux en fonction du nombre de visiteurs et de zones.

Le musée est visité en moyenne par un million de visiteurs tous les ans. Il propose environ 250 zones d'intérêt. Uniquement une minorité des visiteurs visite plusieurs complexes et/ou plusieurs bâtiments.

Q41. Conclure quant à la faisabilité de ce calcul. Justifier une simplification des calculs en segmentant les visites d'après la structure complexe/bâtiment/étage/salle/zone.

La réalisation de cette corrélation met en évidence un nombre limité de profils-types pour lesquels les personnels du musée proposent des contenus ciblés.

L'application exécutée sur le smartphone du visiteur mesure en temps réel sa trace, calcule l'empreinte et cherche le profil le plus adapté parmi ceux disponibles, par corrélation. Cette opération est réalisée par la fonction « meilleur_profil » dont la déclaration est la suivante :

```
constexpr size_t NB_ZONES 256; // nombre de zones considérées
typedef array<seconds,NB_ZONES> empreinte; // occupation par zone
int meilleur_profil(empreinte visiteur, vector<empreinte> profils);
```

Avec :

- « visiteur » liste tous les temps écoulés dans chacune des zones pour le visiteur,
- « profils » liste toutes les empreintes des profils-types, chaque profil liste les temps écoulés dans chacune des zones
- la valeur de retour est l'indice du profil le plus ressemblant (pour lequel le résultat de la corrélation est maximal)

Q42. Implémenter la fonction « meilleur_profil »

Q43. Conclure sur la nécessité de vérifier les trajectoires obtenues et sur la validité des méthodes proposées. Proposer d'autres méthodes de complétions automatisées des trajectoires incomplètes.

DOSSIER TECHNIQUE

Le dossier technique comporte huit documents techniques :

Tournez la page S.V.P.

Document technique DT1. Attributs de la classe ScanResult

D'après https://developer.android.com/reference/android/net/wifi/ScanResult

```
public final class ScanResult extends Object
```

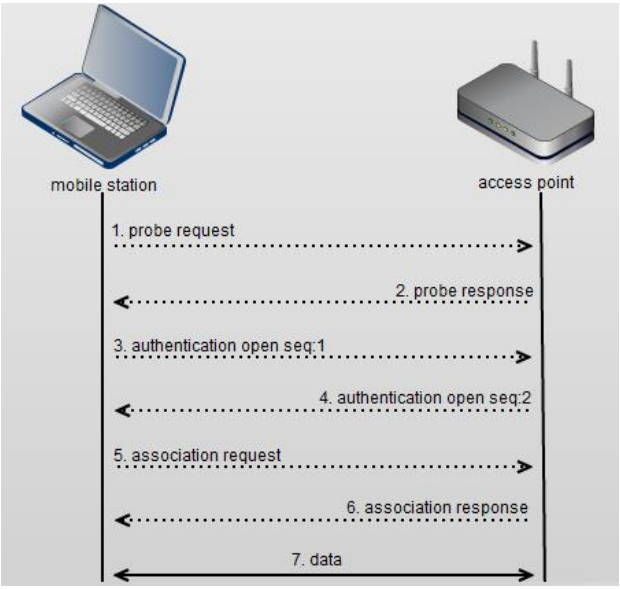
Fields	
public String	BSSID The MAC address of the access point.
public String	SSID The network name.
public String	capabilities Describes the authentication, key management, and encryption schemes supported by the access point.
public int	centerFreq0 Not used if the AP bandwidth is 20 MHz If the AP use 40, 80 or 160 MHz, this is the center frequency (in MHz) if the AP use 80 + 80 MHz, this is the center frequency of the first segment (in MHz)
public int	centerFreq1 Only used if the AP bandwidth is 80 + 80 MHz if the AP use 80 + 80 MHz, this is the center frequency of the second segment (in MHz)
public int	channelWidth AP Channel bandwidth; one of CHANNEL_WIDTH_20MHZ, CHANNEL_WIDTH_40MHZ, CHANNEL_WIDTH_80MHZ, CHANNEL_WIDTH_160MHZ or CHANNEL_WIDTH_80MHZ_PLUS_MHZ.
public int	frequency The primary 20 MHz frequency (in MHz) of the channel over which the client is communicating with the access point.
public int	level The detected signal level in dBm, also known as the RSSI.
public long	timestamp timestamp in microseconds (since boot) when this result was last seen.

Document technique DT2. Association Wi-Fi

D'après
https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/802.11_Association_Process_Explained

The three 802.11 connection states are:

- Not authenticated or associated.
- Authenticated but not yet associated.
- Authenticated and associated.



A mobile station starts out as not authenticated and associated.

1. A mobile station randomizes its MAC address and sends probe requests to discover 802.11 networks within its proximity. Probe requests advertise the mobile stations supported data rates and 802.11 capabilities such as 802.11n. Because the probe request is sent from the mobile station to the destination layer-2 address and BSSID of ff:ff:ff:ff:ff:ff all AP's that receive it will respond.

2. APs receiving the probe request check to see if the mobile station has at least one common supported data rate. If they have compatible data rates, a probe response is sent advertising the SSID (wireless network name), supported data rates, encryption types if required, and other 802.11 capabilities of the AP.

A mobile station chooses compatible networks from the probe responses it receives. Compatibility could be based on encryption type. Once compatible networks are discovered the mobile station will attempt low-level 802.11 authentication with compatible APs. Keep in mind that 802.11 authentication is not the same as WPA2 or 802.1X authentication mechanisms which occur after a mobile station is authenticated and associated. Originally 802.11 authentication frames were designed for WEP encryption however this security scheme has been proven to be insecure and therefore deprecated. Because of this 802.11 authentication frames are open and almost always succeed.

3. A mobile station sends a low-level 802.11 authentication frame to an AP setting the authentication to open and the sequence to 0x0001.

4. The AP receives the authentication frame and responds to the mobile station with authentication frame set to open indicating a sequence of 0x0002.

If an AP receives any frame other than an authentication or probe request from a mobile station that is not authenticated it will respond with a deauthentication frame placing the mobile into an unauthenticated an unassociated state. The station will have to begin the association process from the low level authentication step. At this point the mobile station is authenticated but not yet associated. Some 802.11 capabilities allow a mobile station to low-level authenticate to multiple APs. This speeds up the association process when moving between APs. A mobile station can be 802.11 authenticated to multiple APs however it can only be actively associated and transferring data through a single AP at a time.

5. Once a mobile station determines which AP it would like to associate to, it will send an association request to that AP. The association request contains chosen encryption types if required and other compatible 802.11 capabilities.

If an AP receives a frame from a mobile station that is authenticated but not yet associated, it will respond with a disassociation frame placing the mobile into an authenticated but unassociated state.

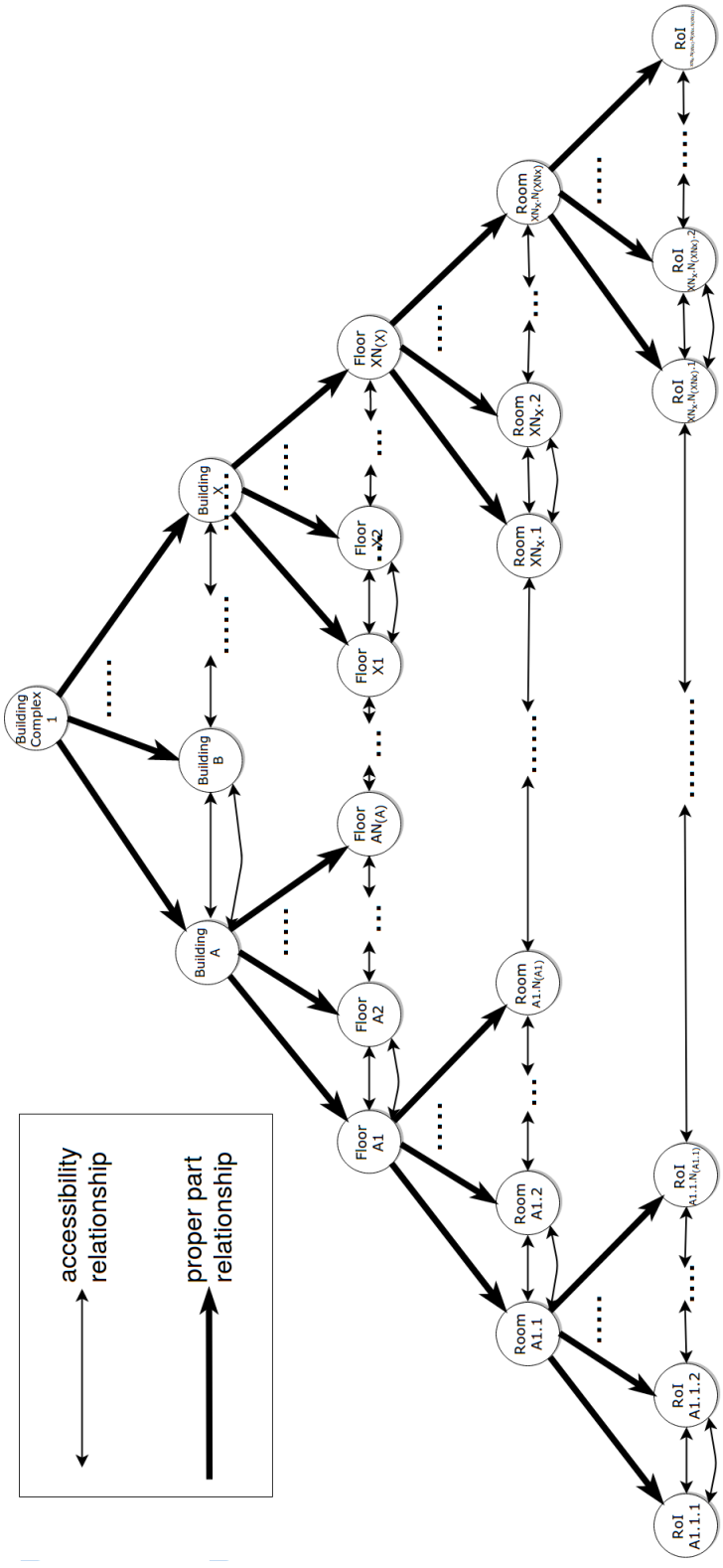
6. If the elements in the association request match the capabilties of the AP, the AP will create an Association ID for the mobile station and respond with an association response with a success message granting network access to the mobile station.

7. Now the mobile station is successfully associated to the AP and data transfer can begin.

Note: If WPA/WPA2 or 802.1X authentication is required on the wireless network, the mobile station will not be able to send data until dynamic keying and authentication have taken place after the 802.11 Association is complete.

Document technique DT3. Structure des localisations

La figure ci-dessous est extraite de l'article : Kontarinis, A., Zeitouni, K., Marinica, C. *et al.* Towards a semantic indoor trajectory model: application to museum visits. *Geoinformatica* **25,** 311–352 (2021). https://doi.org/10.1007/s10707-020-00430-x



Tournez la page S.V.P.

Document technique DT4. Eléments de syntaxe SQL « CREATE TABLE »

Pour les éléments de syntaxe ci-dessous :

- les mots en majuscules sont des mots clés du langage
- les mots en gras sont des identificateurs choisis par le programmeur,
- les mots en italiques sont des références à d'autres éléments de syntaxe,
- les parenthèses ( ) font partie de la commande,
- les accolades { } limitent la portée des opérateurs suivants,
- les crochets [ ] entourent une portion facultative de la syntaxe,
- les points de suspension … indiquent une répétition 1 ou + du bloc précédent,
- la barre verticale | indique un choix entre plusieurs éléments de syntaxe.

La syntaxe simplifiée de création d'une table correspond aux éléments suivants :

```
CREATE TABLE [IF NOT EXISTS] tbl_name (create_definition
[create_definition , …])

create_definition: col_name column_definition | index_definition

column_definition: data_type [NOT NULL | NULL] [AUTO_INCREMENT]

index_definition: {
  PRIMARY KEY (index_col_name)
| UNIQUE [INDEX|KEY] [index_name] (index_col_name)
| CONSTRAINT FOREIGN KEY (index_col_name) reference_definition }

reference_definition:
    REFERENCES tbl_name (index_col_name)
      [ON DELETE reference_option]
      [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL
```

Pour l'élément de syntaxe *reference_option*, les définitions des choix possibles sont :

- RESTRICT la modification ou la suppression d'une valeur de la colonne est refusée si une référence existe vers cette valeur (comportement par défaut si *reference_option* est omis),
- CASCADE la modification ou la suppression d'une valeur de la colonne est autorisée et automatiquement propagée aux références vers cette valeur,
- SET NULL si la valeur nulle est autorisée pour toutes les références, la modification ou la suppression d'une valeur de la colonne est autorisée et les éventuelles références sont modifiées comme nulle ; sinon la modification ou la suppression est refusée.

Document technique DT5. Eléments de syntaxe SQL « CREATE VIEW »

La syntaxe simplifiée de création d'une vue correspond aux éléments suivants :

```
CREATE [OR REPLACE] [DEFINER = { user | CURRENT_USER }] [SQL SECURITY {
DEFINER | INVOKER }] VIEW view_name [(column_list)] AS select_statement

column_list: column_name [, column_name …]

select_statement: SELECT select_expr [, select_expr ...]
     FROM table_references WHERE where_condition
select_expr: column_name | table_name.column_name

table_references: table_name [, table_name …]
```

where_condition: toute expression sur les données ayant un résultat booléen

Tournez la page S.V.P.

Document technique DT6. Manipulations des flux et instants

Les exemples commentés ci-dessous sont écrits en C++14 :

```
#include <chrono>   // types seconds, minutes, jours ainsi que suffixes
#include <sstream>  // chaînes de caractères sous la forme de flux
#include <iostream> // flux standard cout vers la console d'exécution

using namespace std; // std::string peuvent être raccourci en string
using namespace std::chrono; // idem avec chrono::seconds…
// le type seconds a la documentation suivante :
// seconds = duration</*signed integer type of at least 35 bits*/>

// converti une chaîne de caractères HH:MM:SS en secondes
seconds seconds_from_string(string& str)
{
    istringstream is{str}; // flux en lecture seule pour lire str
    char unused; // pour extraire les caractères ":"
    unsigned int h,m,s;
    is >> h;       // lecture des heures
    is >> unused;  // élimination du séparateur
    is >> m;       // lecture des minutes
    is >> unused;  // élimination du séparateur
    is >> s;       // lecture des secondes
    // calcul et retour du nombre équivalent de secondes
    // 1h utilise le suffixe "h", construit un objet "hours"
    // 1min utilise le suffixe "min", construit un objet "minutes"
    // 1s utilise le suffixe "s", construit un objet "seconds"
    return (h * 1h) + (m * 1min) + (s * 1s);
}

int main()
{
    // variable de type "seconds" initialisé par une constante
    seconds sec1 = 15h + 12min + 10s;
    // endl = end-of-line
    cout << "sec1=" << sec1 / 1s << endl; // affiche sec1=54730
    // constante chaîne de caractères
    string s = "15:12:09";
    // conversion de la chaîne de caractères en secondes
    seconds sec2 = seconds_from_string(s);
    cout << "sec2=" << sec2 / 1s << endl; // affiche sec2=54729
    seconds diff = sec1 - sec2;
    cout << "diff=" << diff / 1s << endl; // affiche diff=1
    // « seconds » peut être comparés avec les opérateurs habituels
    // if(sec1 > sec2) …
    return 0; // retour de la fonction main, 0 = pas d'erreur
}
```

Document technique DT7. Manipulations des tableaux

Les exemples commentés ci-dessous sont écrits en C++14 :

```
#include <iostream> // flux standard cout vers la console d'exécution
#include <array>    // définition des tableaux fixes
using namespace std; // std::array disponible simplement comme array

int main()
{
    // création d'un tableau de 5 entiers préinitialisés (facultatif)
    array<int,5> a{1,10,100,200,2};
    cout << "taille=" << a.size() << endl; // affiche taille=5
    // accès en lecture à un élément du tableau
    cout << "a[1]=" << a[1] << endl; // affiche a[1]=10
    int sum = 0;
    // itération sur toutes les valeurs du tableau (de 0 à size()-1)
    for(auto it = a.begin(); it != a.end(); it++)
        sum += *it; // accès à l'élément itéré (écriture possible)
    cout << "somme=" << sum << endl; // affiche Somme=313
    cout << "a[2]=" << a[2] << endl; // affiche a[2]=100
    a[2] = 18; // modification d'un élément d'un tableau
    cout << "a[2]=" << a[2] << endl; // affiche a[2]=18
    return 0; // retour de la fonction main, 0 = pas d'erreur
}
```

```
#include <iostream> // flux standard cout vers la console d'exécution
#include <vector>   // définition des tableaux dynamiques
using namespace std; // std::vector disponible simplement comme vector

int main()
{
    // création d'un tableau de 2 entiers préinitialisés (facultatif)
    vector<int> v{2,5};
    cout << "taille=" << v.size() << endl; // affiche taille=2
    cout << "v[1]=" << v[1] << endl; // affiche v[1]=5
    int prod = 1;
    // itération sur toutes les valeurs du tableau (de 0 à size()-1)
    for(auto it = v.begin(); it != v.end(); it++)
        prod *= *it; // accès à l'élément itéré (écriture possible)
    cout << "produit=" << prod << endl; // affiche produit=10
    cout << "v[1]=" << v[1] << endl; // affiche v[1]=5
    v[1] = 10; // modification d'un élément du tableau
    cout << "v[1]=" << v[1] << endl; // affiche v[1]=10
    v.push_back(10); // ajout d'un élément à la fin du tableau
    cout << "taille=" << v.size() << endl; // affiche taille=3
    cout << "v[2]=" << v[2] << endl; // affiche v[2]=10
    return 0; // retour de la fonction main, 0 = pas d'erreur
}
```

Document technique DT8. Indicateurs par étage

Le code ci-dessous est écrit en C++20 (pour l'utilisation de ranges::fill) :

```
#include <chrono>     // définitions et calculs sur le temps
#include <algorithm>  // pour ranges::fill
#include <vector>     // pour les tableaux de type vector
using namespace std;
using namespace std::chrono;
// déclaration de annotation, presence et trace issues du DR4
constexpr size_t MAX_ETAGE = 100; // équivalent C++ du #define

void occupation_etage(
    vector<trace> trajectoires, // toutes déjà chargées en mémoire
    vector<seconds>& tcumul, // temps de cumul pour tous les étages
    vector<int>& vcumul) // visiteurs pour tous les étages
{
    // l'identifiant d'un étage est toujours entre 0 et MAX_ETAGE-1
    tcumul.reserve(MAX_ETAGE); // un élément par étage
    // remise à zéro de tous les temps cumulés pour tous les étages
    ranges::fill(tcumul.begin(), tcumul.end(), 0s);

    vcumul.reserve(MAX_ETAGE); // un élément par étage
    // remise à zéro de toutes les présences pour tous les étages
    ranges::fill(vcumul.begin(), vcumul.end(), 0);

    // les vecteurs sont correctement initialisés à zéro et peuvent
    // à présent stocker les cumuls pour tous les étages

    // pour toutes les trajectoires
    for(auto itt = trajectoires.begin();
        itt != trajectoires.end(); itt++)
    {
        // trajectoire courante
        trace& trajectoire = *itt;

        // pour toutes les présences de la trajectoire courante
        for(auto itp = trajectoire.begin();
            itp != trajectoire.end(); itp++)
        {
            // présence courante
            presence& p = *itp;

            // cumul du temps de visite pour cet étage
            tcumul[p.E] += p.tend - p.tstart;

            // cumul du nombre d'utilisateur pour cet étage
            vcumul[p.E] += 1;
        }
    }
}
```

DOSSIER REPONSE

Le dossier réponse comporte cinq documents réponses :

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR1 à DR3

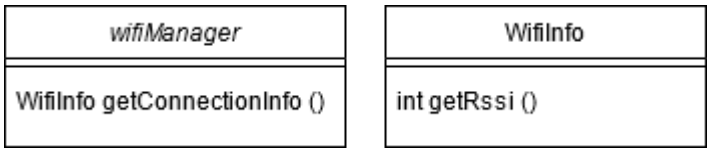
Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR1. Lecture niveau de réception RSSI

Question Q7. Compléter le code ci-dessous en langage proche de JAVA pour récupérer le niveau de réception *RSSI* dans une variable nommée « rssi ».

Les diagrammes partiels des classes disponibles sont :




```
Object service = context.getSystemService(Context.WIFI_SERVICE);

WifiManager wifiManager = (WifiManager) service;
```

Question Q8. Opération réalisée par « WifiManager wifiManager = (WifiManager) service; » :

Question Q8. Diagramme de classes correspondant :

Document réponse DR2. Coloriage d'une zone

Question Q12. Compléter le pseudocode de la fonction « coloriage »

L'objectif de cet algorithme est de parcourir tous les vecteurs de déplacement d'une matrice et de chercher les similitudes entre voisins. Le résultat est une matrice colorée :

- le fond de la vidéo, immobile, est caractérisé par la couleur (valeur numérique) 0,
- chaque objet identifié correspond à une couleur (valeur numérique) unique.

Cette matrice colorée est également stockée sous la forme d'un vecteur de 12288 éléments.

```
Fonction coloriage(   // colorie un objet de voisin en voisin
     Depl est un vecteur de 12288 n-uplets (x,y),
     Obj est un vecteur de 12288 couleurs,
     j est le point de départ du coloriage,
     Coul est la couleur de remplissage de l'objet
     // coloriage de la case courante

     (x,y) ← Depl(j)
     // x correspond au déplacement horizontal de la case courante
     // y correspond au déplacement vertical de la case courante
     Si(                          ) // la case de gauche existe
          (xg,yg) ← Depl(             )
          // si non colorié et même déplacement
          Si(                                  )
               // récursivité vers la gauche

          Fin Si
     Fin Si
     // coloriage vers la droite non-détaillé, semblable à la gauche
     Si(                          ) // la case du dessous existe
          (xb,yb) ← Depl(             )
          // si non colorié et même déplacement
          Si(                                  )
               // récursivité vers le bas

          Fin Si
     Fin Si
     // coloriage vers le haut non-détaillé, semblable au bas
Fin Fonction
```

Document réponse DR3. Coloriage d'un objet

Question Q13. Compléter le pseudocode d'appel de la fonction « coloriage »

Cette fonction « coloriage » est appelée par le code suivant :

```
Depl est un vecteur de 12288 n-uplets (x,y)
Res est un vecteur de 12288 couleurs, tout initialisé à 0, à calculer
Coul est un entier, initialisé à 0, à calculer
Pour i=0 jusqu'à 12287 avec un pas de 1,
      (x,y) ← Depl(i)
      Si(                                          )
            // non colorié encore et déplacement est non-nul
                                  // allocation d'une nouvelle couleur
            // appel de la fonction de coloriage


      Fin si
Fin pour
```

Question Q16. Compléter le pseudocode d'estimation de la validité et de la position moyenne de chaque objet.

```
Res est un vecteur de 12288 couleurs (entier), déjà calculées
Sx est un vecteur d'entiers (somme des positions x pour chaque objet)
Sy est un vecteur d'entiers (somme des positions y pour chaque objet)
N est un vecteur d'entiers (nombre d'éléments pour chaque objet)
Sx, Sy, N, V sont initialisés à zéro, à calculer
Pour i=0 jusqu'à 12287 avec un pas de 1,
      (x,y) ← Coord(i) // calcul coordonnée de l'élément courant









Fin pour
Mx, My sont des vecteurs de flottants
V est un vecteur de booléens (validité pour chaque objet)
Pour i=0 jusqu'à 255 avec un pas de 1,









Fin pour
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR4 - DR5

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR4. Déclaration des types associés

Question 30. Définir la structure associée à la présence datée dans une zone du musée, les champs seront nommés « Z », « S », « E », « B », « C », « tstart », « tend » et « A » :

```
// annotation est une structure opaque, définie ultérieurement
typedef struct {} annotation;

typedef struct
{









} presence;
```

Question 30. Définir le type « trace » comme étant un tableau de « presence » :

```
typedef                              trace;
```

Question 33. Compléter le code de la fonction « corrige_trace » :

```
// t représente la trace à vérifier/corriger
// la fonction n'a volontairement pas de valeur de retour
void corrige_trace(trace& t)
{
// t[i] ou *it permet d'accéder à une présence
// t[i].tstart ou it->tstart pour accéder aux champs d'une présence
// les éventuels indices sont à déclarer avec le type « size_t »
// les éventuels itérateurs sont à déclarer avec le type « auto »









}
```

Document réponse DR5. Interpolation des présences

Question 34. Compléter le code de la fonction « interpole_trace » :

```
// pour une zone donnée
// renvoie la salle S, l'étage E, le bâtiment B et le complexe C
void details_zone(int Z, int& S, int& E, int& B, int& C);
// renvoie la liste de zones à interpoler entre les zones données
// cette liste est vide si les zones Z1 et Z2 sont adjacentes
vector<int> interpole_zone(int Z1, int Z2);

trace interpole_trace(const trace t)
{
      // cette trace vide est à compléter avec les éléments de t,
      trace resultat; // éventuellement intercalés d'éléments interpolés
// t[i] ou *it permet d'accéder à une présence
// t[i].Z ou it->Z pour accéder aux champs d'une présence
// les éventuels indices sont à déclarer avec le type « size_t »
// les éventuels itérateurs sont à déclarer avec le type « auto »


      // la première présence est toujours ajoutée au résultat
      resultat.push_back(*it0);
      for(
      {
            vector<int> Z = interpole_zone(
            // si aucune zone à interpoler
            if(
            {
                  // interpolation du temps dans chaque zone
                  seconds tdelta = (          -          ) /         ;
                  seconds tstart = it0->tend; // temps de début
                  // pour tous les éléments du vecteur Z
                  for(
                  {
                        presence p; // champs non-initialisés sauf A vide
                        p.Z =
                        // récupération des détails de la zone
                        details_zone(
                        // interpolation du temps d'entrée

                        // interpolation du temps de sortie


                        // ajout de la présence interpolée

                  }
            }
            // ajout de la présence suivante

      }
      return resultat;
}
```