Université de Nantes
Master 2 Bioinformatique
2019 – 2020

Nom
Prénom
(Rabattre l'angle s.v.p.)

Contrôle continu
Advanced algorithmics and programming for biologists

C. Sinoquet
Mardi 17 décembre 2019
16H00-17H30

documents de cours, TD et TP autorisés
tout autre document interdit

Nombre d'intercalaires supplémentaires utilisés :

Vous devez répondre sur le présent document. En cas de manque de place, veuillez indiquer le numéro d'intercalaires supplémentaires utilisés en débutant à 1. Le présent document n'est pas compté comme intercalaire. Si aucune référence ne figure, la réponse ne sera pas lue.

Les copies illisibles ou comportant des fautes d'orthographe seront sanctionnées.

Toute réponse doit être justifiée (éventuellement par un schéma). Une réponse oui/non ne rapportera aucun point.

Vous devez répondre en utilisant les structures algorithmiques de pseudo-code ou de C/C++ vues en cours ou en Travaux Dirigés. Une syntaxe « à la Python » n'est pas autorisée.

ENONCE

Question 1 – Transformation directe d'une liste chaînée en liste chaînée comportant un élément sur deux

```
exemple :

liste initiale : 1 2 3 4 5 6
liste initiale après transformation : 1 3 5

La transformation doit être faite directement sur la liste initiale, dont on élimine une cellule sur deux. On ne doit pas générer de nouvelle liste en se guidant sur la liste initiale puis supprimer la liste initiale.
```

On définit les types suivants :

TCell : type structure comportant les champs
- info : de type entier
- next : de type pointeur sur TCell

TPointer : type pointeur sur TCell

Q1.1 Complétez par e(entrée), s(sortie), es(entrée/sortie) la signature suivante, donnée en pseudo-code :

PROCEDURE reduction_liste(….tete : TPointer)

Q1.2. Ecrivez le pseudo-code de la procédure suivante :
PROCEDURE reduction_liste(….tete : TPointer)

Question 2 – Fusion de deux listes de messages horodatés, triées par ordre croissant sur le champ horodatage

On souhaite fusionner les deux listes en une troisième liste, en conservant le tri sur le champ horodatage et en rajoutant une information sur la liste d'origine.

Pour les deux listes à fusionner, on considère les types suivants :

TMess1 : type structure comportant les champs
horodatage : de type entier
message : de type chaîne de caractères TChar80
next : de type pointeur sur TMess1

TPointer1 : type pointeur sur TMess1

Pour la nouvelle liste à créer, on considère les types suivants :

TMess2 : type structure comportant les champs
horodatage : de type entier
message : de type chaîne de caractères TChar80
emetteur : de type entier
next : de type pointeur sur TMess2

TPointer2 : type pointeur sur TMess2

```
Exemple :
Première liste : (103,''appel'') (210,''connexion'') (510,''problème résolu'')
Deuxième liste : (300,''appel'') (510,''connexion'') (700,''problème résolu'')

Troisième    liste :    (103,''appel'',1)    (210,''connexion'',1)    (300,''appel'',2)
(510,''problème résolu'',1) (510,''connexion'',2) (700,''problème résolu'',2)
```

Q2.1. En C++, complétez la signature pour les procédure et fonction dont les pseudo-codes sont les suivants :

PROCEDURE fusion_listes_messages_horodates
(… tete1 : TPointer1, ... tete2 : TPointer1, ... tete3 : TPointer2) où
- tete1 désigne la tête de la première liste,
- tete2 désigne la tête de la deuxième liste,
- tete3 désigne la tête de la troisième liste.

FONCTION fusion_listes_messages_horodates_bis
(… tete1 : TPointer1, ... tete2 : TPointer1) : …...

Q2.2. En C++, écrivez le code de la procédure fusion_listes_messages_horodates.

Q2.3. En C++, écrivez un appel à la procédure fusion_listes_messages_horodates, dans une fonction main.

Question 3 – Adaptation de l'algorithme de tri quick_sort pour trier sur un champ particulier lorsque les données décrivent plusieurs champs.

On utilise les structures de données suivantes :

vect1 : vecteur d'entiers enregistrant les poids moléculaires de molécules chimiques
vect2 : vecteur d'entiers enregistrant les numéros de référence de ces molécules

Ainsi, par exemple, vect1[6] et vect2[6] concernent la même molécule.

Q3.1. En pseudo-code, adaptez l'algorithme de tri quick_sort étudié en cours pour trier par ordre croissant vect1 et garder vect2 cohérent. Il y a deux algorithmes à écrire.

La signature de l'algorithme adapté est :

```
PROCEDURE tri_quick_sort(es vect1 : vecteur d'entiers,
                         es  vect2 : vecteur d'entiers,
                         e   g : entier,
                         e   d : entier)
```

qui fait appel à

```
PROCEDURE separer(es  vect1 : vecteur d'entiers,
                  es  vect2 : vecteur d'entiers,
                  e   g : entier,
                  e   d : entier,
                  s   indice_pivot : entier)
```

Q3.2. En C++, pour gérer correctement la modification des vecteurs vect1 et vect2, la signature de la procédure tri_quick_sort ci-dessous est-elle correcte ou doit-elle être modifiée ? Justifiez votre réponse.

```
void tri_quick_sort(vect1 : vector<int>,
                    vect2 : vector<int>,
                    g :     int,
                    d :     int)
```

Question 4 - Parcours en profondeur d'abord d'un arbre quelconque avec comptage du nombre total des nœuds présents dans l'arbre

Q4.1. Définissez le type TNode permettant à un nœud d'avoir un nombre de fils non déterminé à l'avance, sans gaspillage de mémoire. Pour définir TNode, vous pouvez faire appel à d'autres types définis dans le présent document, ou à définir.

Q4.2. Ecrivez en C++ l'algorithme de parcours en profondeur d'abord avec comptage du nombre de nœuds, correspondant à la signature :

PROCEDURE parcours_comptage_noeuds(e n : TTree, es nb_noeuds : entier)

avec TTree : type pointeur sur TNode

Q4.3. Ecrivez en C++ l'appel (en fonction main) à la procédure écrite en Q4.2.

Q4.4. Ecrivez la trace des appels à parcours_comptage_noeuds, pour l'arbre suivant :