Nom :

Prénom :

Login EPITA :

UID :

Examen d'algorithmique

EPITA ING1 2010 Rattrapage, A. DURET-LUTZ

Durée : 1 heure 30

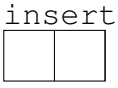
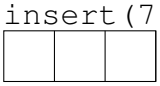
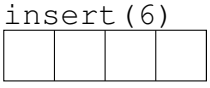
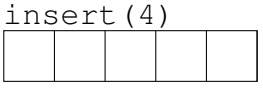
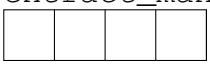
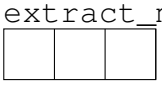
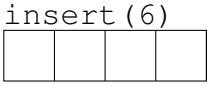
Avril 2008

Consignes

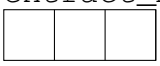
- Tous les documents (sur papier) sont autorisés, livres inclus.
  Les calculatrices, téléphones, PSP et autres engins électroniques ne le sont pas.
- Répondez sur le sujet dans les cadres prévus à cet effet.
- Il y a 7 pages. Rappelez votre UID en haut de chaque feuille au cas où elles se mélangeraient.
- Ne donnez pas trop de détails. Lorsqu'on vous demande des algorithmes, on se moque des points-virgules de fin de commande etc. Écrivez simplement et lisiblement. Des spécifications claires et implémentables sont préférées à du code C ou C++ verbeux.
- Le barème est indicatif et correspond à une note sur 20.

File (2 points)

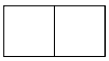
Dans cette question on considère une file de priorité S implémentée à l'aide d'un tas « max », c'est-à-dire avec la valeur maximale à la racine du tas.

1. S est initialement vide. Donnez l'état du tableau représentant S après y avoir effectué chacune des opérations suivantes :
   - `insert(5)`
     
   - `insert(2)`
     
   - `insert(7)`
     
   - `insert(6)`
     
   - `insert(4)`
     
   - `extract_max()`
     
   - `extract_max()`
     
   - `insert(6)`
     

- `extract_max()`



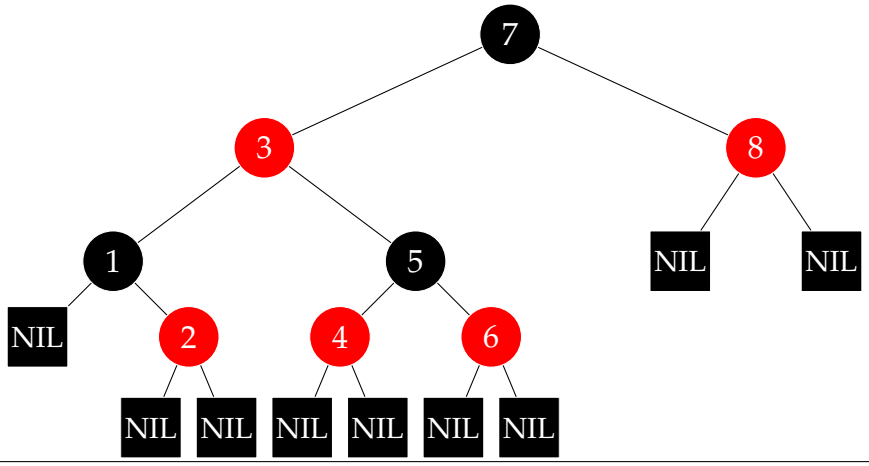
- `extract_max()`



Identification d'Arbres Rouge et Noir (3 points)

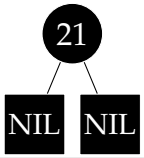
Pour chacun des arbres qui suivent, indiquez si ce sont des arbres binaires de recherche rouge et noir. Si l'arbre n'est pas un arbre rouge et noir, précisez pourquoi. (Les négations non justifiées seront ignorées.)
Pour décoder les photocopies en noir et blanc, voici du rouge et voici du noir.

1.



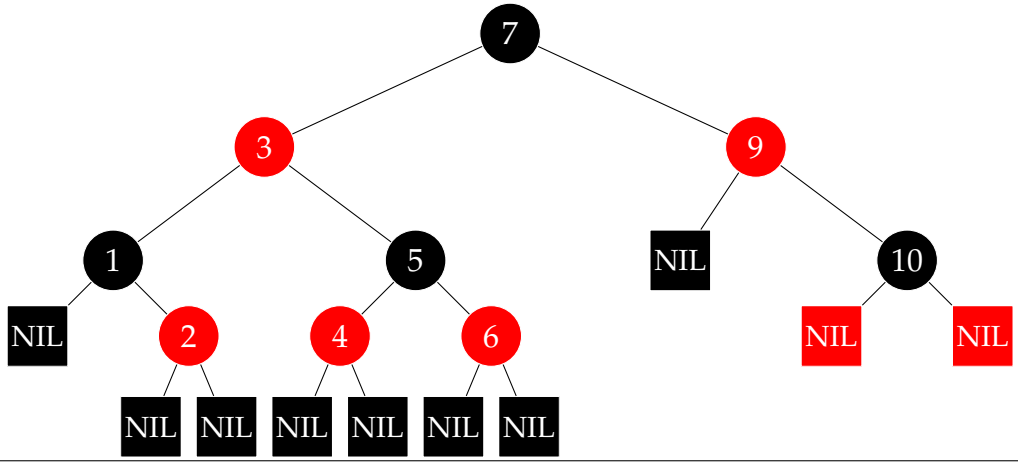
Réponse :

2.



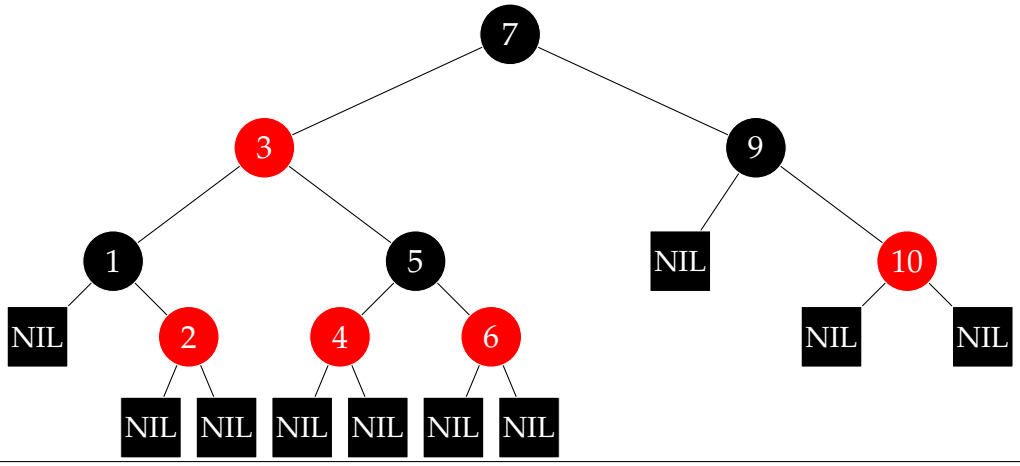
Réponse :

3.



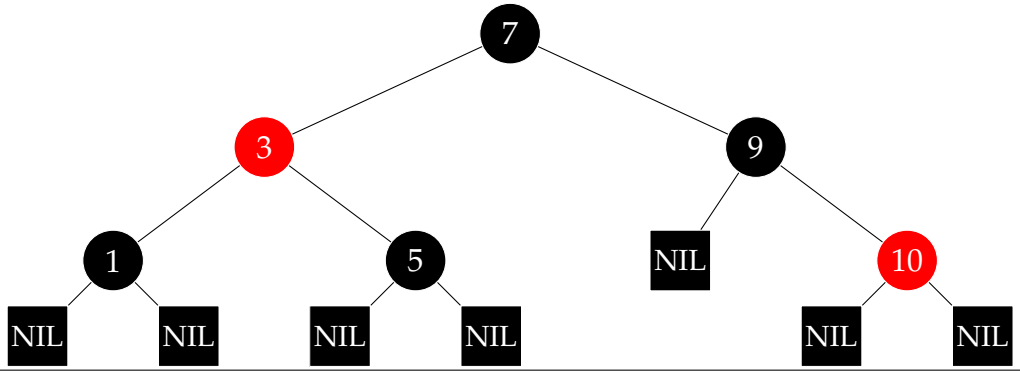
Réponse :

4.



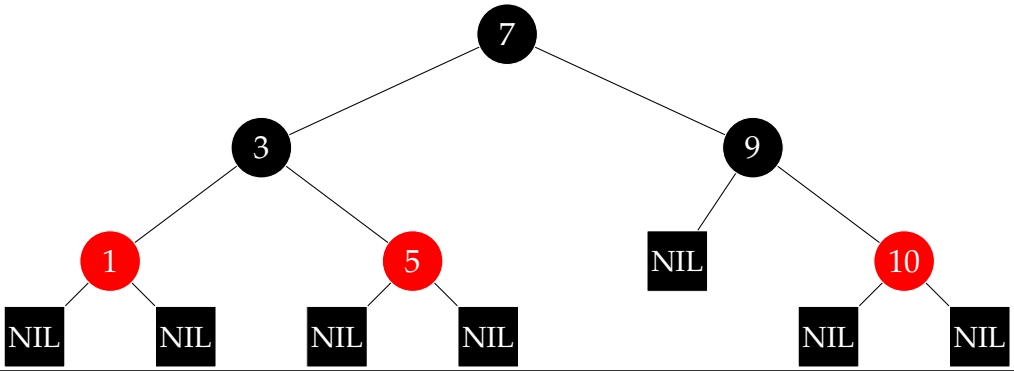
Réponse :

5.



Réponse :

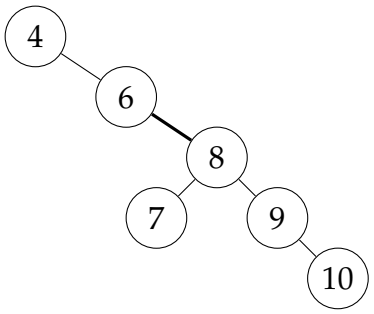
6.



Réponse :

Rotation d'arbre binaire de recherche (1 point)

Redessinez l'arbre binaire de recherche suivant après avoir effectué une rotation gauche de l'arc (6)—(8).



Réponse :

Complexités (3 points)

1. **(1 point)** Ai-je le droit d'écrire $\Theta(n) - \mathrm{O}(n) = \Theta(n)$? Justifiez votre réponse mathématiquement.

   Réponse :

2. **(2 point)** La complexité $T(n)$ d'un algorithme est constante pour une entrée de taille $n \leq 2$ et vérifie la relation $T(n) = 2T(n/3) + n \log n$ pour $n > 2$. Que pouvez-vous dire de $T(n)$?

   Réponse :

Comb sort (5 points)

Le « *comb sort* » ou « tri à peigne » est une amélioration du tri à bulles qui prétend rivaliser en vitesse avec des tris plus sérieux comme le quicksort.

Le tri à bulles fonctionne en plusieurs passes qui parcourent le tableau entièrement de haut en bas : lors de chaque passe, chaque élément du tableau est comparé avec l'élément précédent, ces deux éléments sont éventuellement permutés pour faire remonter les petites valeurs (les bulles) et descendre les grosses. Ces passes se répètent jusqu'à ce qu'aucune permutation ne soit nécessaire. Le problème du tri à bulles et qu'il peut exister des « tortues » : c'est-à-dire des petites valeurs vers la fin du tableau qui vont demander beaucoup de passes pour atteindre leur place finale vers le début du tableau.

Le *comb sort* supprime les tortues en modifiant l'écart entre les éléments comparés. Dans le tri à bulles cet écart est toujours de 1 puisqu'un élément toujours est comparé avec le précédant. Le *comb sort*, en revanche, commence avec un écart aussi grand que la taille du tableau à trier, puis divise cet écart par 1,3 (en arrondissant à l'entier inférieur) après chaque passe. Les grands écarts du début permettent de faire vite remonter les tortues ; à la fin, quand l'écart est devenu 1, le comportement est similaire à un tri à bulles. Le *comb sort* se termine lorsqu'une passe avec un écart de 1 n'a fait aucune comparaison.
Voici une implémentation de l'algorithme, tel qu'il est présenté sur wikipedia :

```
function combsort11(array input)
    gap := input.size //initialize gap size

    loop until gap <= 1 and swaps = 0
        //update the gap value for a next comb
        if gap > 1
            gap := gap / 1.3
            if gap = 10 or gap = 9
                gap := 11
            end if
        end if

        i := 0
        swaps := 0

        //a single "comb" over the input list
        loop until i + gap >= array.size
            if array[i] > array[i+gap]
                swap(array[i], array[i+gap])
                swaps := 1
            end if
            i := i + 1
        end loop
    end loop
end function
```

1. **(4 points)** Donnez et justifiez la complexité temporelle de la fonction `combsort11` dans **le meilleur des cas**, en fonction de la taille n du tableau à trier. Prenez garde au fait que la boucle interne de l'algorithme effectue de plus en plus de comparaisons au fur et à mesure que la variable `gap` décroît.

Réponse :

2. **(1 point)** Cet algorithme de tri est-il stable ou instable ? Pourquoi ?

Réponse :

Le pot (6 points)

L'organigrame de la société ProgDyn est un arbre dont le PDG est la racine. Chaque employé est représenté par un nœud dans cet arbre et, à l'exception du PDG, les employés possèdent tous un supérieur hiérarchique (leur père dans l'arbre) avec lequel ils s'entendent mal.

Pour fêter la découverte d'un nouvel algorithme, la société ProgDyn veut organiser un pot. Pour une meilleure ambiance, un employé ne devra pas y être invité avec son supérieur hiérarchique.

D'autre part chaque employé possède une « note de convivialité » (valeur positive ou nulle).

L'objectif est de trouver l'ensemble des employés à inviter de façon à maximiser la somme des notes de convivialité, sans inviter à la fois un employé et son supérieur.

Notations :
- On note $e(x)$ l'ensemble des employés dont le supérieur est x (les subordonnés immédiats).
- On note $n(x) \geq 0$ la note de convivialité d'un employé x.
- On note p le PDG.

On souhaite résoudre ce problème par programmation dynamique en considérant des sous-arbres de la hiérarchie de taille croissante.

Notons $A[x]$ la somme des notes de convivialité maximale que l'on peut atteindre en choisissant des invités uniquement parmi les subordonnés (immédiats ou non) de x, en invitant x.

Notons $S[x]$ la somme des notes de convivialité maximale que l'on peut atteindre en choisissant des invités uniquement parmi les subordonnés (immédiats ou non) de x, mais sans inviter x.

1. **(2 points)** Donnez des définitions récursives de $A[x]$ et $S[x]$.

Réponse :

2. **(2 points)** En fonction du nombre m d'employés, quelle serait la complexité du calcul de $A[p]$ par programmation dynamique ? (Justifiez votre réponse si vous n'avez pas répondu à la question précédente.)

Réponse :

3. **(2 points)** Une fois que l'on possède un algorithme de programmation dynamique pour calculer $A[p]$, comment le modifier pour obtenir l'ensemble des employés à inviter ?

Réponse :

La base tordue (2 points)

Ne lisez et répondez à cette question, calculatoire, que si vous vous ennuyez ferme. Elle ne fait pas partie du barème.

On considère des nombres formés uniquement des chiffres 0 et 1, mais interprétés dans la base $-1+i$ (où i désigne malheureusement ce que vous croyez : l'unité imaginaire dont le carré vaut -1). Appelons cette base tordue la base T. Par exemple le nombre 110 en base T représente $1 \times (-1+i)^2 + 1 \times (-1+i)^1 + 0 \times (-1+i)^0 = -1-i$ en base 10. De même vous pourrez vérifier que le nombre $3+2i$ est représenté par 1001 en base T.

Cette base permet de représenter de façon unique tous les *nombres complexes entiers*, c'est-à-dire les nombres complexes dont les parties imaginaires et réelles sont entières. (Cette affirmation demanderait une preuve, vous devrez me croire sur parole.)

La question est la suivante : **quelle est la représentation de $-2-i$ en base T ?**

Indices :

- Si vous essayez tous les codages binaires dans l'ordre : 0, 1, 10, 11, 100, ... cela vous coûtera plus de 200 essais avant de tomber sur la solution.
- Il est préférable d'appliquer un algorithme similaire à celui qu'on utilise habituellement pour afficher un nombre dans une base entière donnée.

Réponse :