POLYTECHNIQUE MONTRÉAL

Solutionnaire examen intra

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Q7	
Total	

Identification de l'étudiant(e)		
Nom :	**Prénom :**	
Signature :	**Matricule :**	**Groupe :**

Sigle et titre du cours		*Groupe*	*Trimestre*
INF2010 – Structures de données et algorithmes		**Tous**	**20201**
Professeur		***Local***	***Téléphone***
Ettore Merlo, responsable – Tarek Ould Bachir, chargé de cours		-	
Jour	***Date***	***Durée***	***Heure***
Mardi	**10 mars 2016**	**2h00**	**13h00**

Documentation	*Calculatrice*	
☐ Toute	☐ Aucune	**Les cellulaires, agendas électroniques et téléavertisseurs sont interdits.**
☒ Aucune	☐ Programmable	
☐ Voir directives particulières	☒ Non programmable	

Directives particulières

Ne posez pas de question durant l'examen. En cas de doute, énoncez clairement toute supposition que vous faites.
Un cahier d'annexe est également fourni. Ne remettez pas le cahier d'annexes.

Bonne chance à tous!

Important

Cet examen contient **7** questions sur un total de **14** pages **(excluant cette page)**

La pondération de cet examen est de **30** **%**

Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☒ oui ☐ non

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Listes (3/20 points)

Considérez l'Annexe 1 définissant la classe `OrderedList` qui implémente l'interface `Iterable`.

1.1) **(1 point)** Donnez l'affichage résultant de l'exécution de la fonction `main` définie dans `OrderedList`. Prêtez bien attention à la fonction `add(AnyType x)` !

```
[ 1 2 3 4 5 6 7 8 9 ]
[ 1 3 5 7 9 ]
```

1.2) **(1 point)** Donnez la complexité asymptotique en meilleur cas et en pire cas de la méthode publique `addAll(AnyType[] items)`. On supposera que la liste `OredredList` est initialement vide et que le paramètre `items` contient *n* éléments. Justifiez votre réponse.

Meilleur cas :

O(*n*), cas où `items` est inversement trié.

Pire cas :

O(n^2), cas où `items` est trié.

1.3) **(1 point)** Donnez la complexité asymptotique en meilleur cas et en pire cas de la fonction `removeEven` (`OrderedList<Integer> list`). On supposera que le paramètre `list` contient *n* éléments. Justifiez votre réponse.

Meilleur cas :

O(*n*), aucun retrait, il faut quand même parcourir la liste.

Pire cas :

O(*n*), tous les éléments sont retirés, l'utilisation d'un itérateur simplifie le code.

Question 2 : Tables de dispersement (4 points/20)

Considérez l'Annexe 2 définissant la classe `LLHashTable` qui implémente une table de dispersement avec résolution de collision par chaînage.

2.1) **(1 point)** La fonction `rehash` est appelée une fois durant l'exécution de la fonction `main` définie dans `LLHashTable`. Quel est l'élément dont l'insertion a provoqué le `rehash` ? Prêtez bien attention à la fonction `insert(AnyType x)` !

33

2.2) **(1 point)** Donnez la valeur du membre `m` (taille du membre `array`) avant et après le rehash.

`m` avant le `rehash` : 4

`m` après le `rehash` : 9

2.3) **(1 point)** Donnez l'affichage résultant de l'exécution de la ligne 8 de la fonction `main` définie dans `LLHashTable`.

```
[ 56 26 ]
```

2.4) **(1 point)** Donnez l'affichage résultant de l'exécution de la ligne 15 de la fonction `main` définie dans `LLHashTable`.

```
[ 56 74 57 33 26 ]
```

Question 3 : Tris en $n \log(n)$ (3 points/20)

On désire exécuter l'algorithme `quicksort` pour trier le tableau ci-après. On considère une valeur `CUTOFF` de 4. Le code source vous est fourni à l'Annexe 3.

Indice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Valeurs	16	15	2	9	4	3	6	13	8	7	10	1	12	11

3.1) **(0.25 point)** Donnez l'état du tableau après la mise à l'écart du pivot lors de la première récursion de QuickSort :

Indice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Valeurs	6	15	2	9	4	3	12	13	8	7	10	1	11	16

3.2) **(0.75 point)** Donnez l'état du tableau après l'exécution du partitionnement de la première récursion de `quicksort` :

Indice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Valeurs	6	1	2	9	4	3	10	7	8	11	12	15	13	16

3.3) **(1 point)** Au total, quel est le nombre de fois que la fonction récursive `quicksort` aura été appelée pour exécuter le tri ? Pour éviter toute ambiguïté, la signature de la fonction est reproduite ci-après :

```
private static <AnyType extends Comparable<? super AnyType>>
void quicksort( AnyType [ ] a, int left, int right )
```

Votre réponse: 5

3.4) **(1 point)** Énumérez tous les pivots identifiés durant l'exécution de l'algorithme `QuickSort` et donnez leur position finale. Laissez les lignes inutilisées vides.

Valeur du pivot : 11 ; Position finale du Pivot : 9

Valeur du pivot : 6 ; Position finale du Pivot : 4

Valeur du pivot : _____ ; Position finale du Pivot :_____

Valeur du pivot : _____ ; Position finale du Pivot :_____

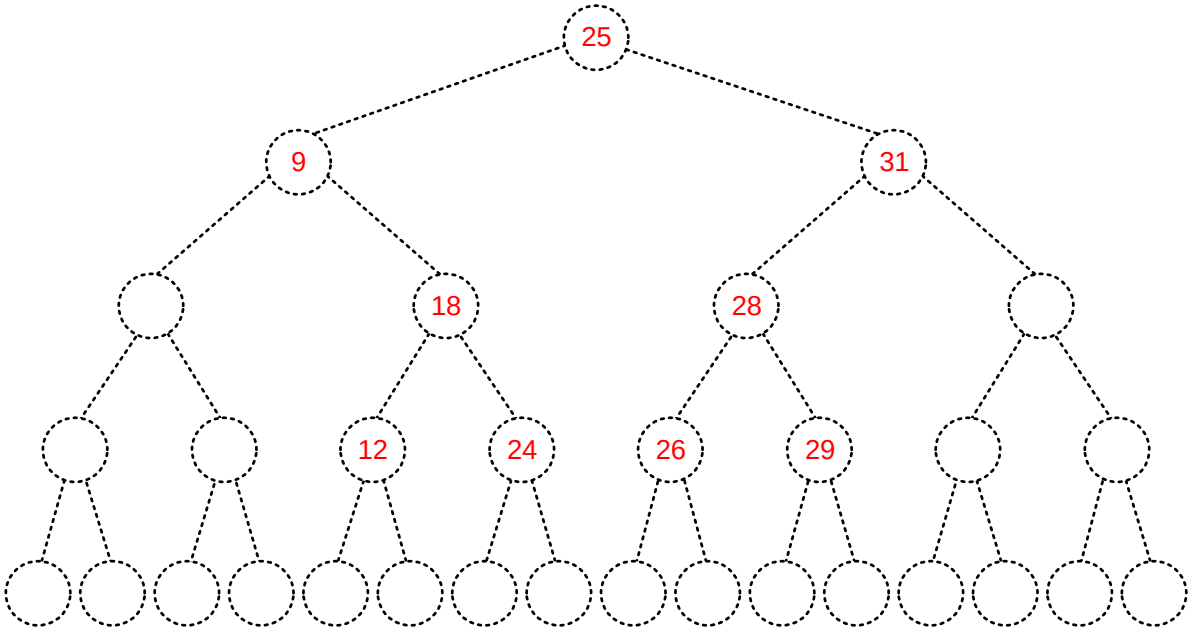
Valeur du pivot : _____ ; Position finale du Pivot :_____

Question 4 : Arbres binaire de recherche (2 points/20)

4.1) **(0.5 point)** Si l'affichage par niveaux de l'arbre binaire de recherche donne :

25, 9, 31, 18, 28, 12, 24, 26, 29

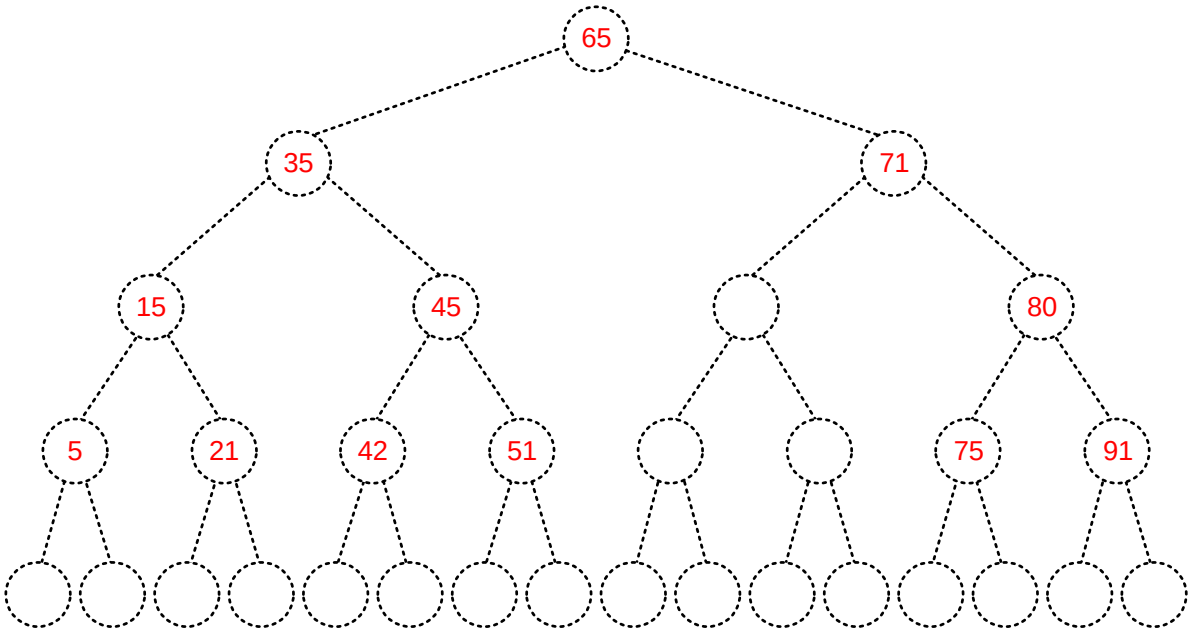
Donnez la représentation graphique de l'arbre.



4.2) **(0.5 point)** Si l'affichage post-ordre de l'arbre binaire de recherche donne :

5, 21, 15, 42, 51, 45, 35, 75, 91, 80, 71, 65

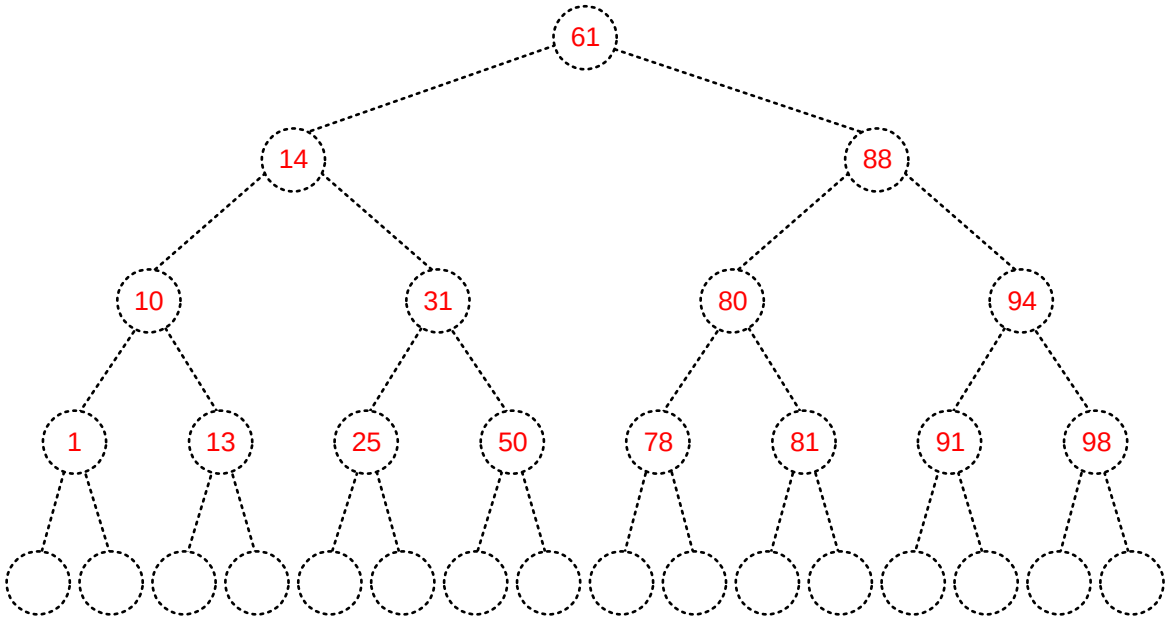
Donnez la représentation graphique de l'arbre.



4.3) **(0.5 point)** Si l'affichage en-ordre de l'arbre binaire de recherche donne :

1, 10, 13, 14, 25, 31, 50, 61, 78, 80, 81, 88, 91, 94, 98

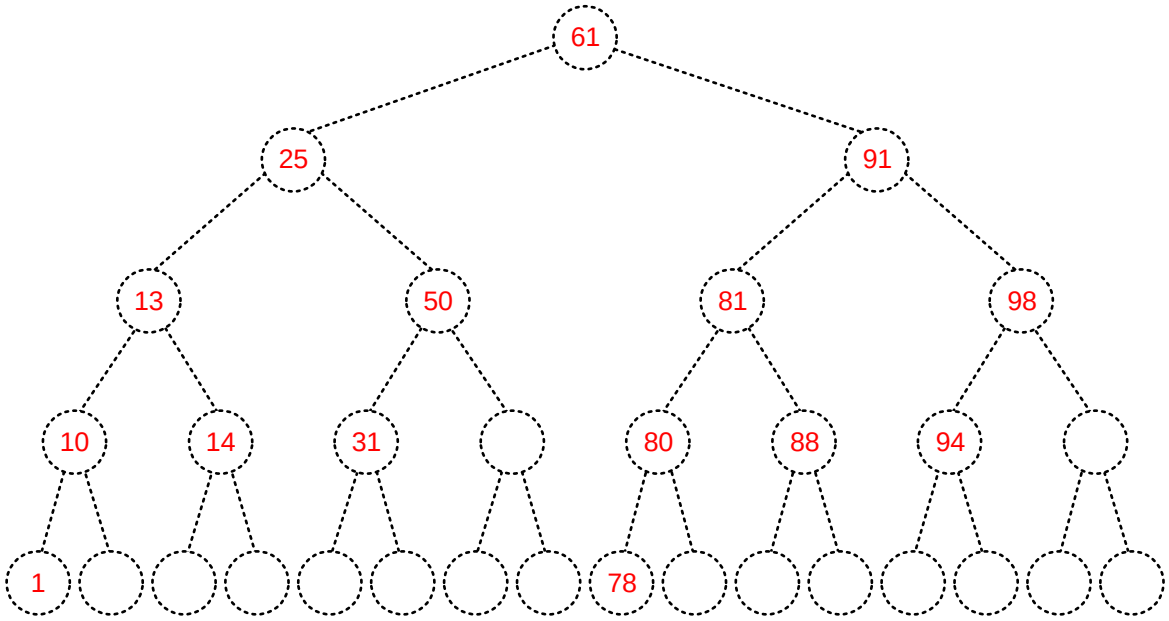
Donnez la représentation graphique de l'arbre, sachant que les sous-arbres à la gauche et à la droite de la racine sont des AVL de hauteur h = 2. Advenant que ces arbres soient débalancés à un de leurs nœuds, le débalancement irait vers la droite.



4.4) **(0.5 point)** Si l'affichage en-ordre de l'arbre binaire de recherche donne :

1, 10, 13, 14, 25, 31, 50, **61**, 78, 80, 81, 88, 91, 94, 98

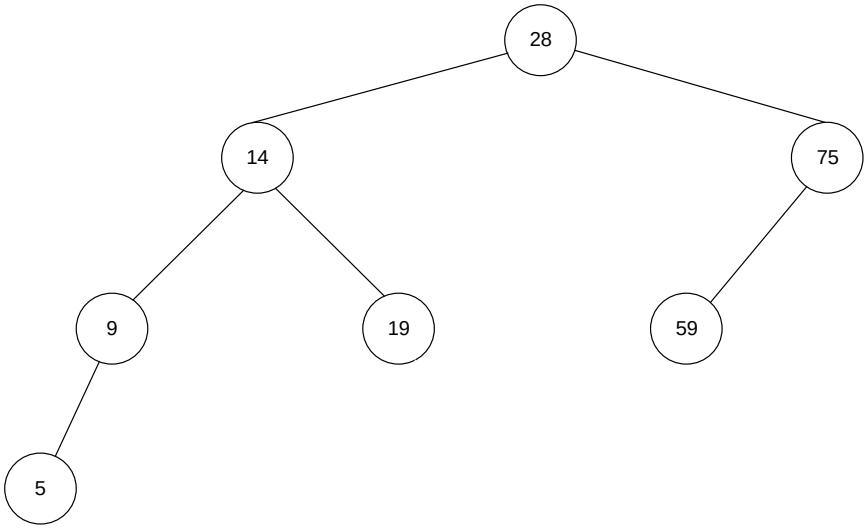
Donnez la représentation graphique de l'arbre, sachant que les sous-arbres à la gauche et à la droite de la racine sont des AVL de hauteur h = 3. Advenant que ces arbres soient débalancés à un de leurs nœuds, le débalancement irait vers la gauche.



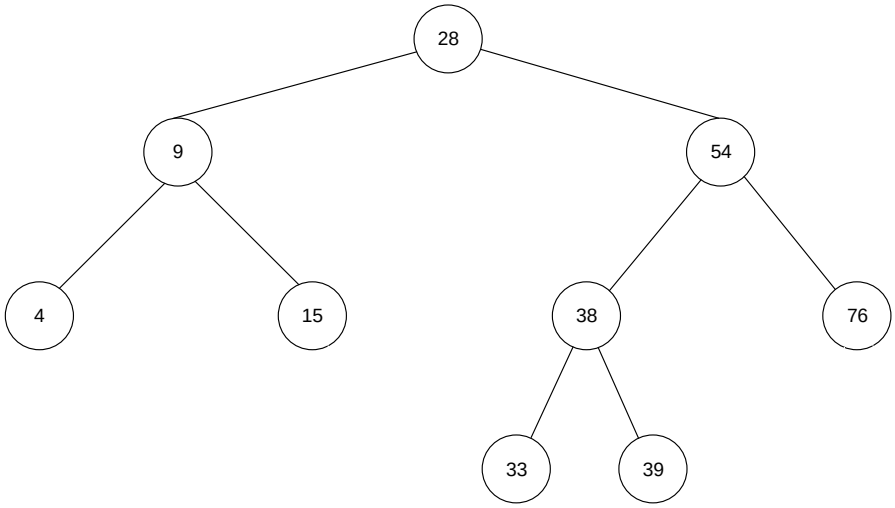
Question 5 : Arbre binaire de recherche de type AVL (3 points)

En partant de chacun des arbres AVL suivants, effectuer les opérations demandées.

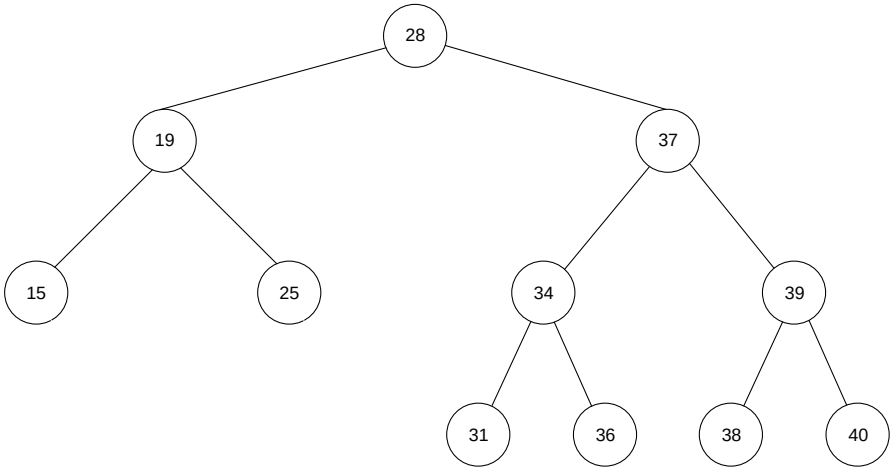
AVL 5A :



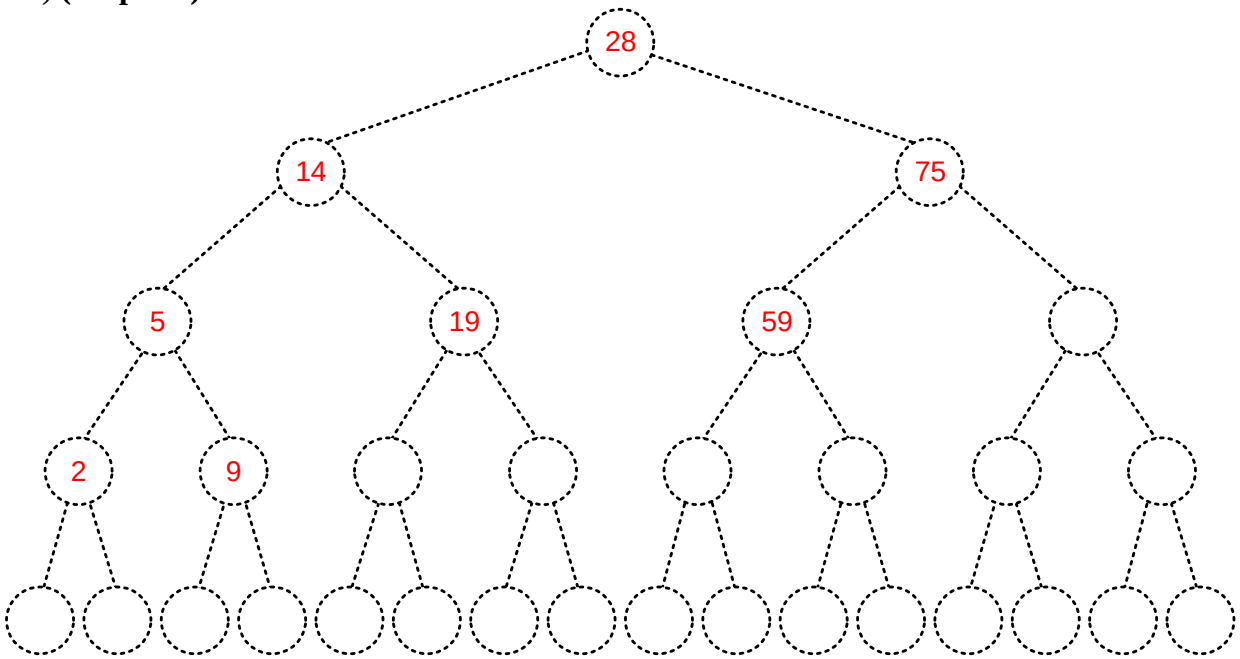
AVL 5B :



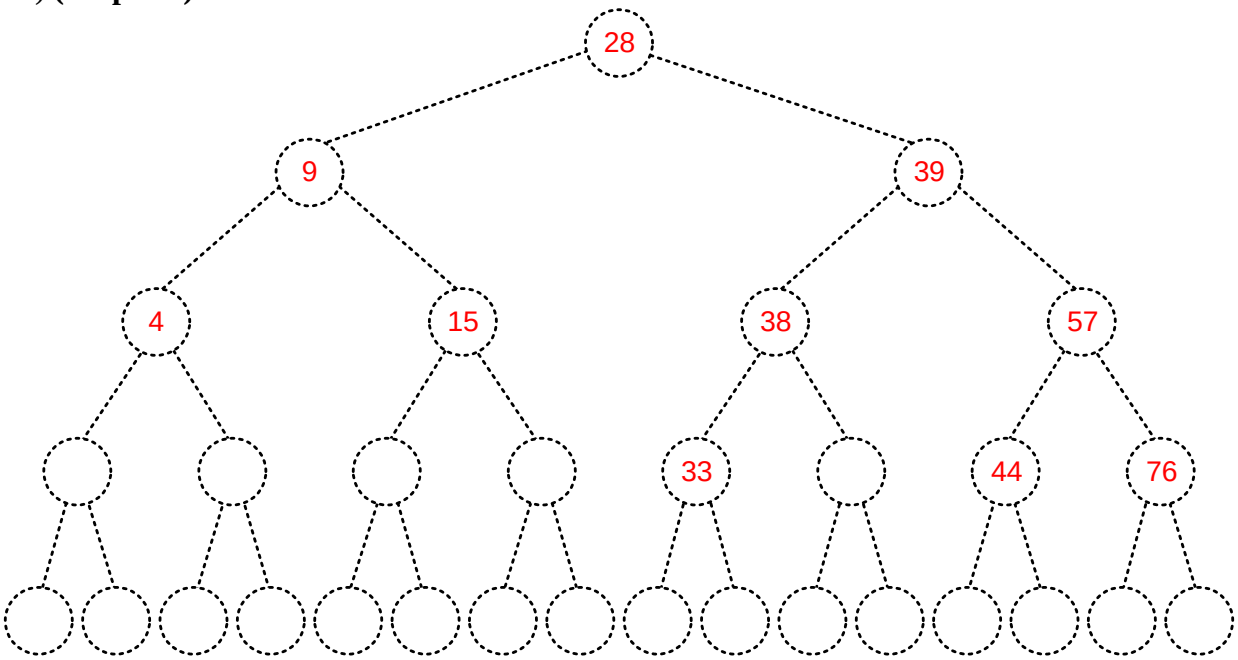
AVL 5C :



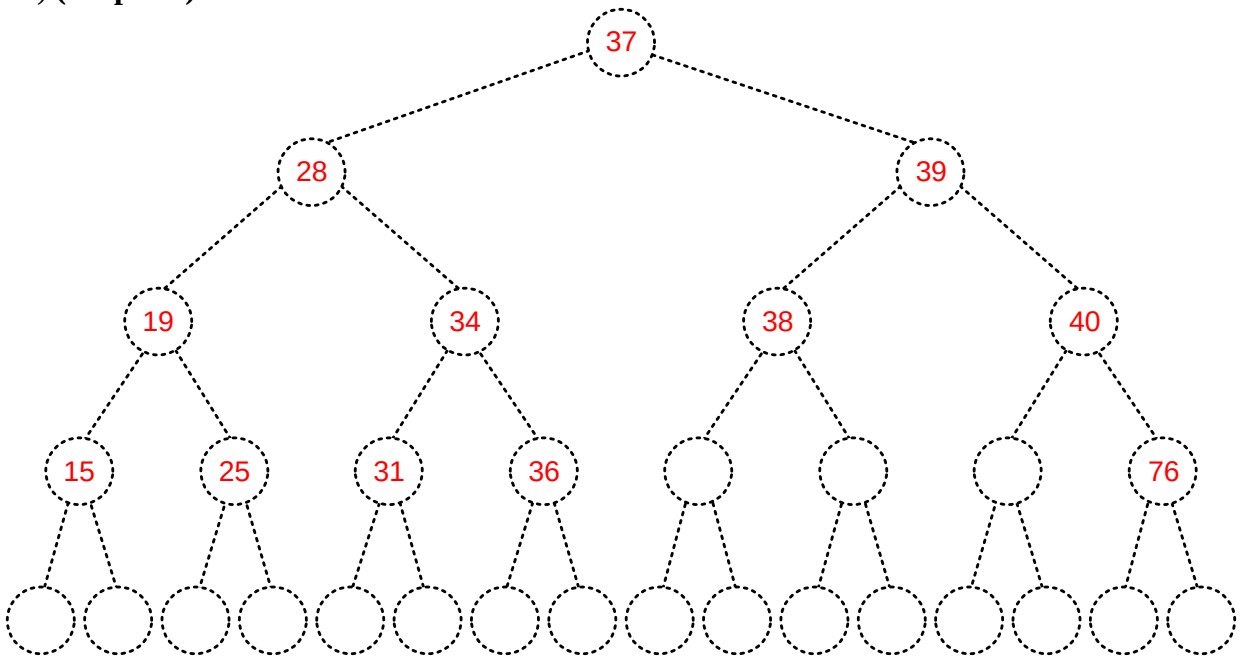
5.1) **(0.5 point)** Insérez 2 à l'AVL 5A.



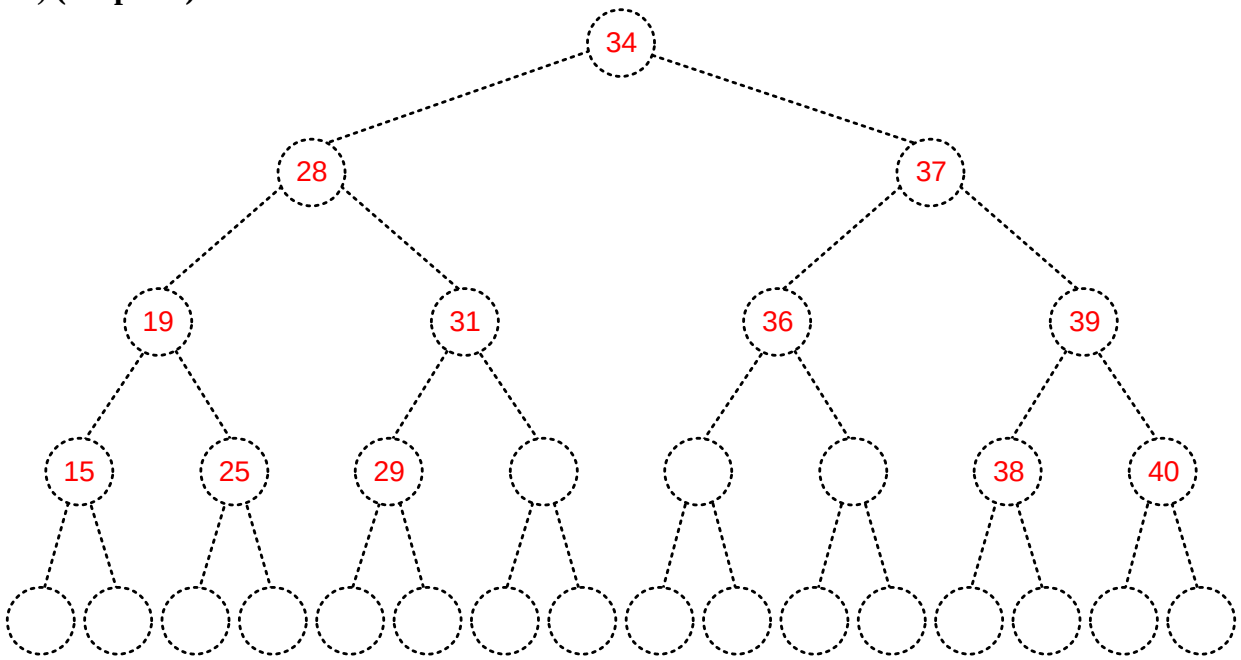
5.2) **(0.5 point)** Insérez 44 à AVL 5B.



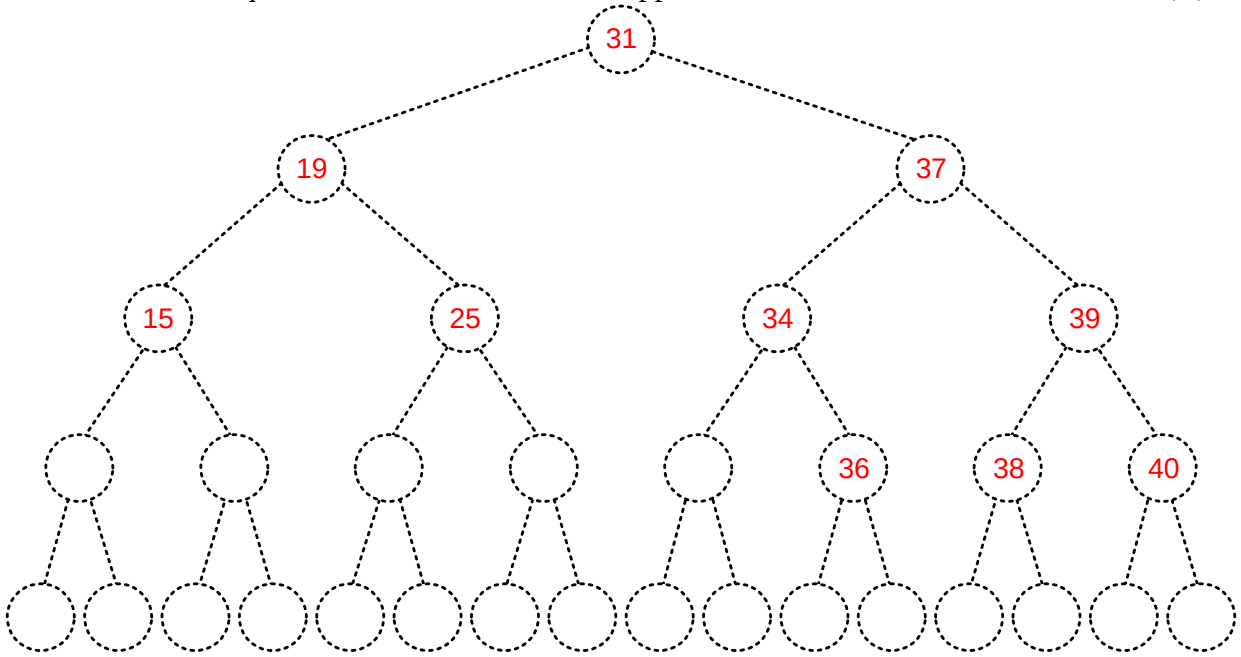
5.3) **(0.5 point)** Insérez 76 à l'AVL 5C.



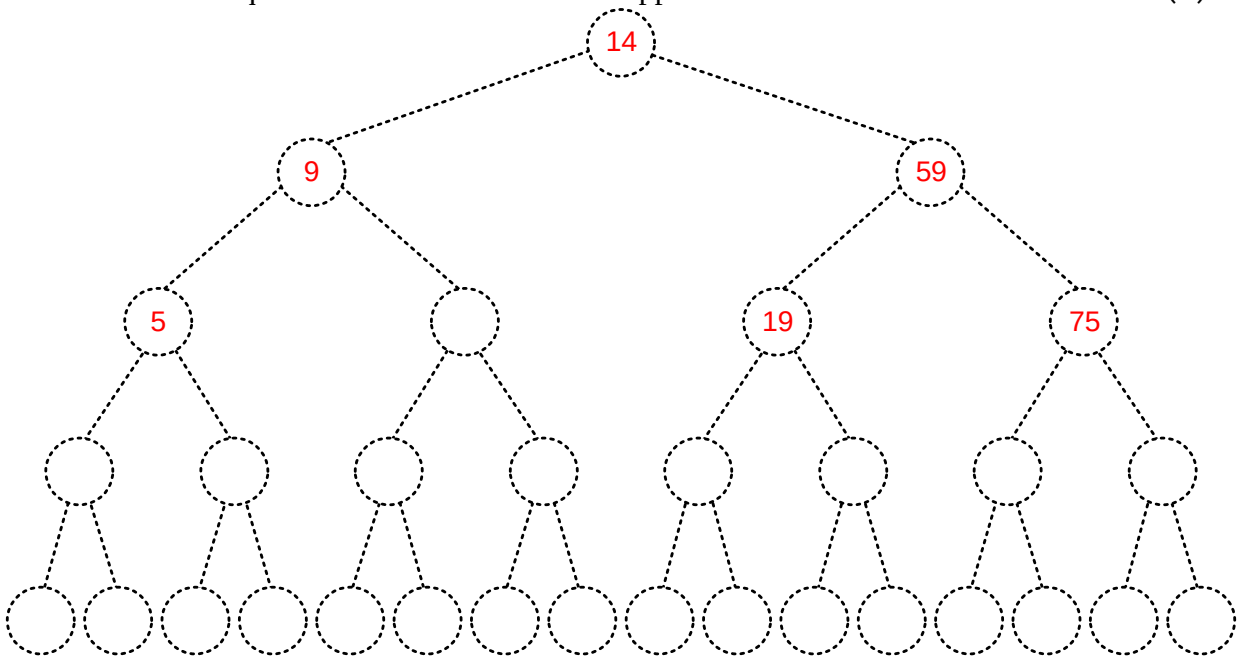
5.4) **(0.5 point)** Insérez 29 à l'AVL 5C.



5.5) **(0.5 point)** Retirez la racine de l'AVL 5C. Aidez-vous de la méthode `remove` donnée à l'Annexe 4. Notez que cette fonction `remove` fait appel à la routine de balancement `balance(t)`.

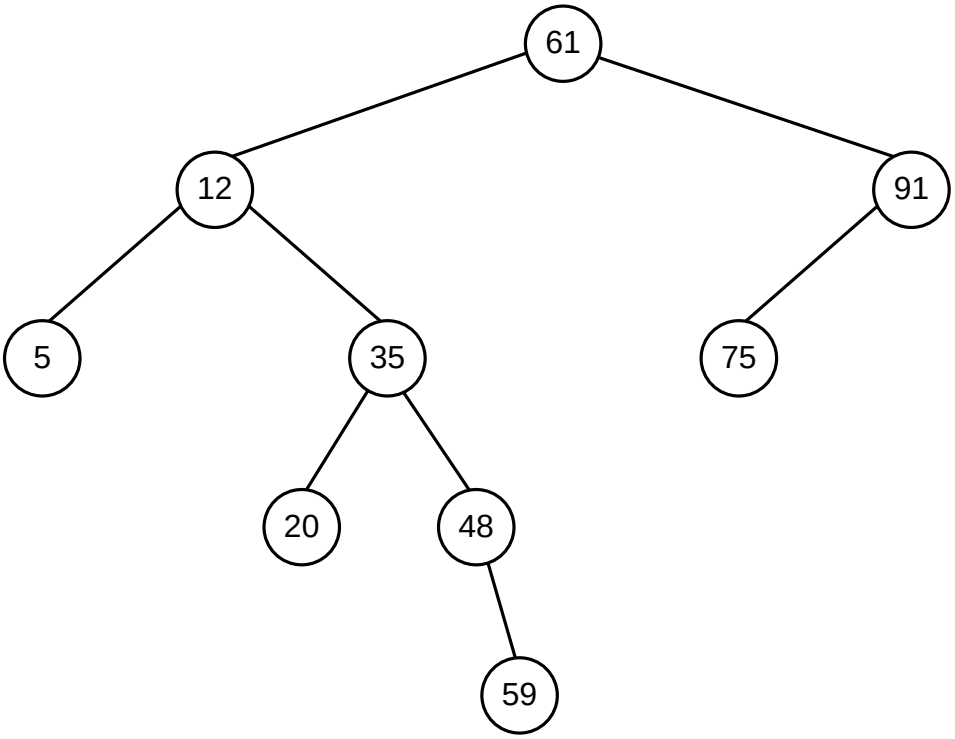


5.6) **(0.5 points)** Reprenez la racine de l'AVL 5A. Aidez-vous de la méthode `remove` donnée à l'Annexe 4. Notez que cette fonction `remove` fait appel à la routine de balancement `balance(t)`.

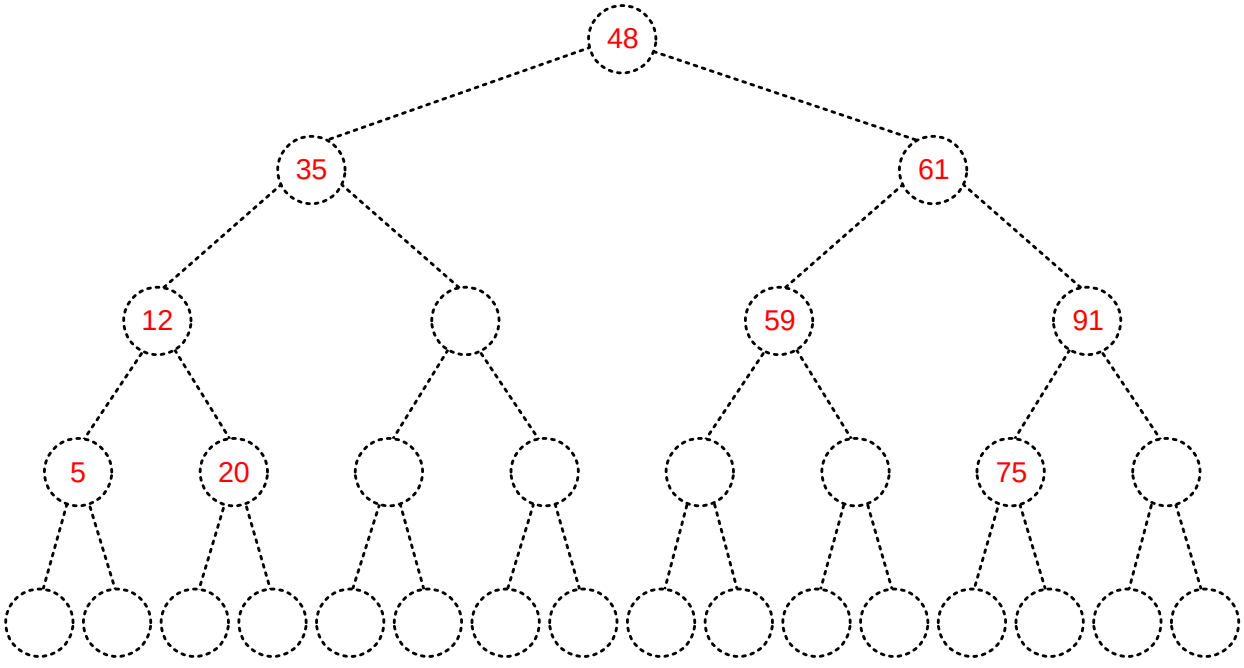


Question 6 : Arbre binaire de recherche de type Splay (3 points/20)

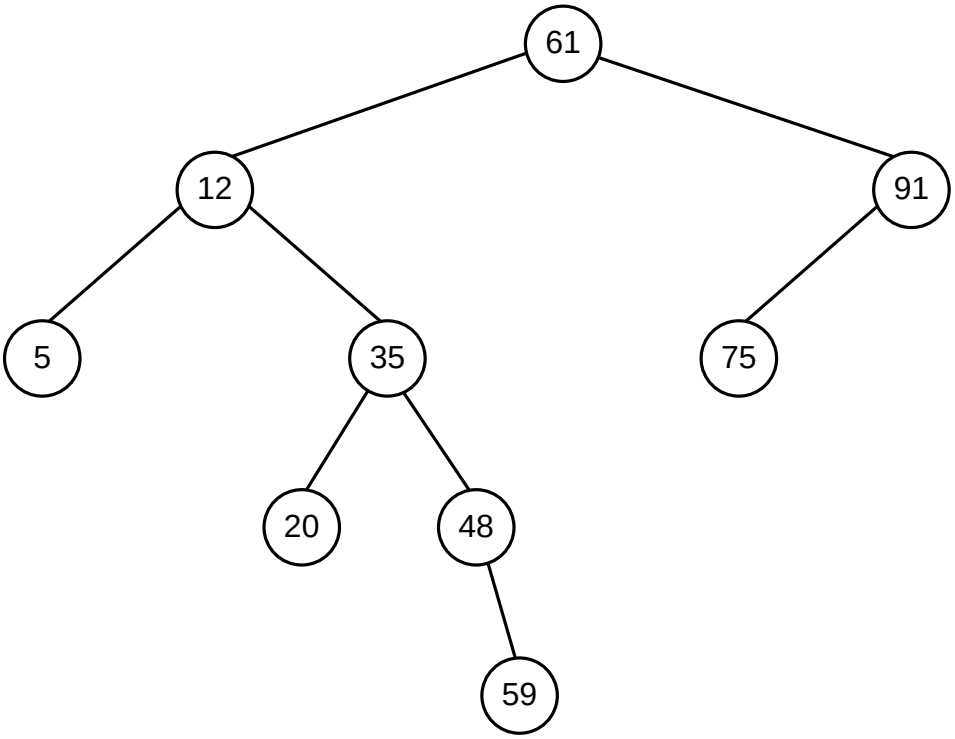
En partant de l'arbre Splay suivant :



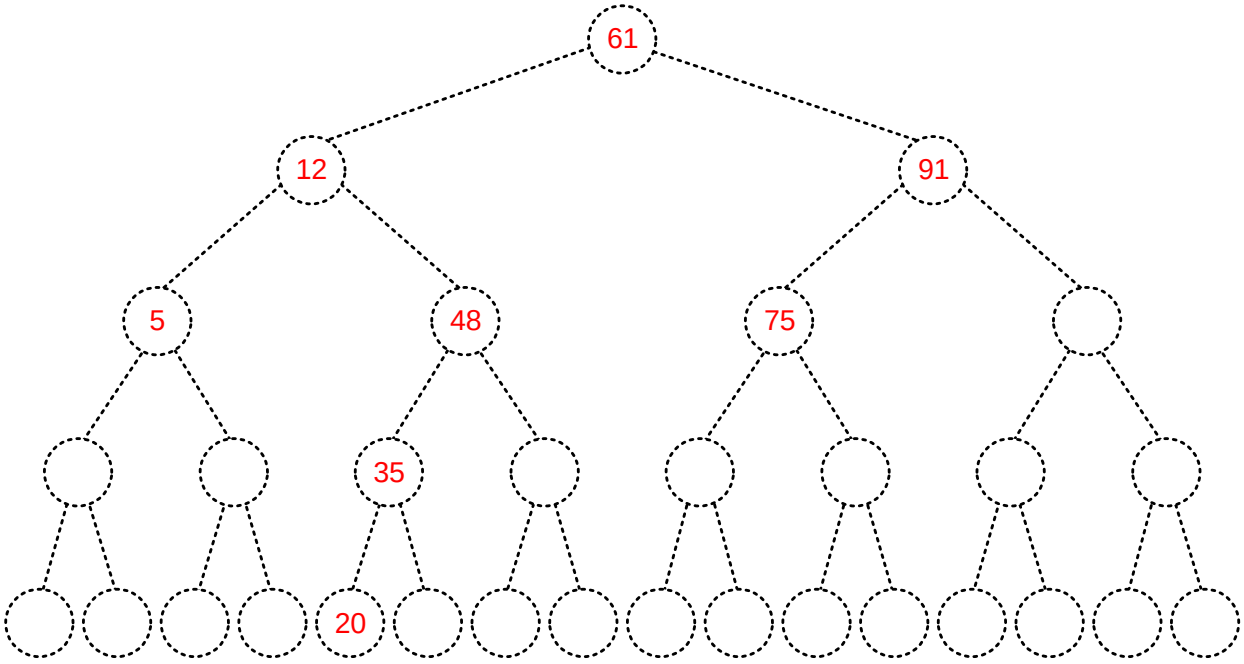
6.1) **(0.5 point)** Effectuez un `get(48)`.



En repartant du même arbre Splay :

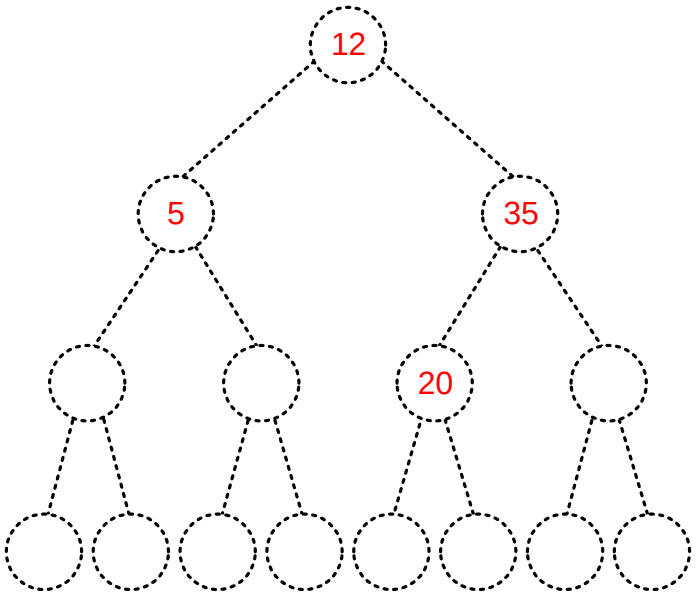


6.2) **(0.5 points)** Effectuez un `delete(59)`.

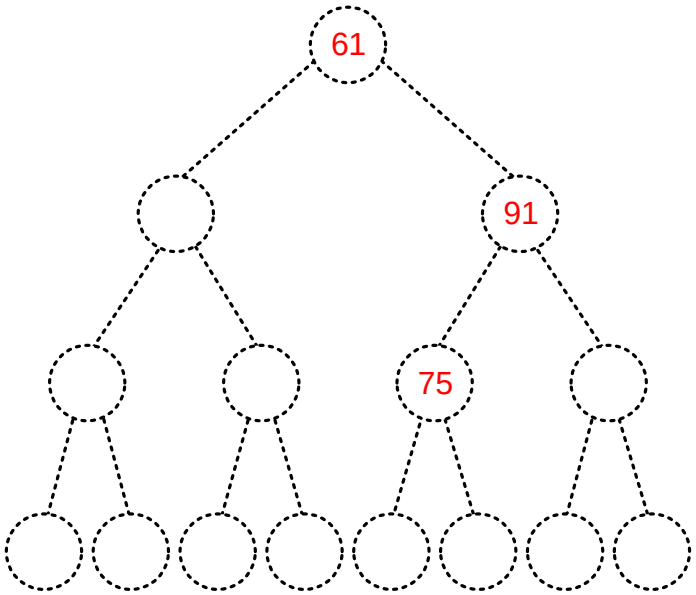


6.3) **(1 point)** Si le `get(48)` de la question 6.1) avait été exécuté par une implémentation de type top-down, quels auraient été les sous-arbres R et L juste avant que 48 ne soit placé à la racine ? **Aidez-vous des données de l'Annexe 5.**

Sous-arbre L:

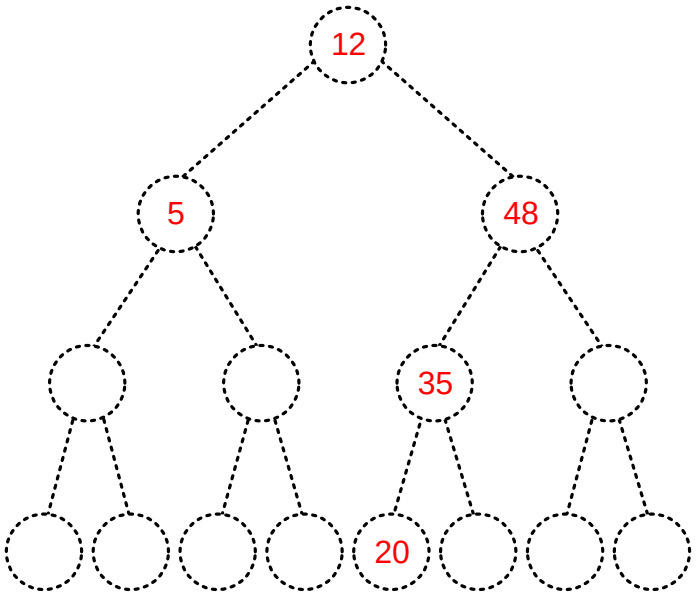


Sous-arbre R:

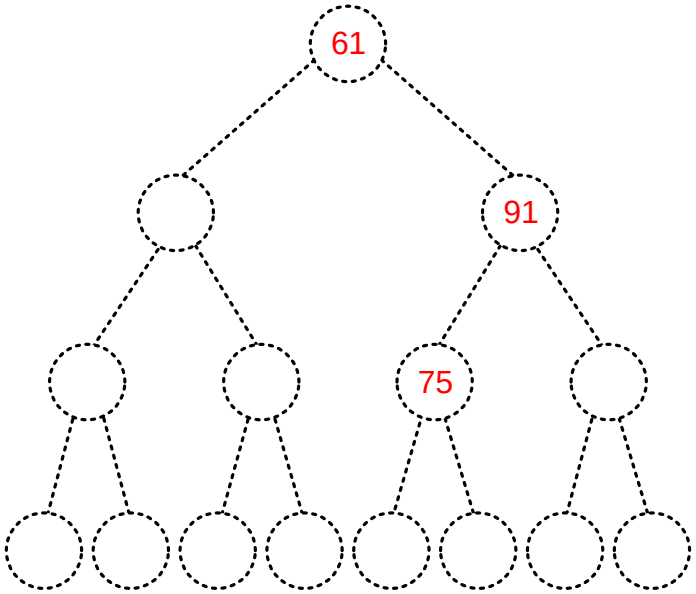


6.4) **(1 point)** Si le `delete(59)` de la question 6.2) avait été exécuté par une implémentation de type top-down, quels auraient été les sous-arbres R et L juste avant que 59 ne soit placé à la racine ? **Aidez-vous des données de l'Annexe 5.**

Sous-arbre L:



Sous-arbre R:



Question 7 : Généralités (2 points/20)

Répondez aux assertions suivantes par « vrai » ou par « faux ». Il ne vous est pas demandé de justifier votre réponse. Les mauvaises réponses sont cependant sanctionnées (pointage négatif) ! Par conséquent, si vous ne connaissez pas la réponse à une question, ne répondez pas !

7.1) **(0.5 point)** En pire cas, l'algorithme MergeSorte a une complexité O($n \log(n)$)

VRAI	X
FAUX	

7.2) **(0.5 point)** Il est toujours possible d'insérer un nouvel élément dans une table de dispersement utilisant une résolution de collision par sondage quadratique dont la taille est un nombre premier et dont le facteur de compression est de 50%.

VRAI	X
FAUX	

7.3) **(0.5 point)** Si la liste `lst` est un `ArrayList`, le code suivant aura une complexité $O(n^2)$.

```
public static void retireNombresPaires (List<Integer> lst ) {
   Iterator<Integer> itr = lst.iterator();
   while( itr.hasNext() )
      if( itr.next() % 2 == 0 )
         itr.remove();
}
```

VRAI	X
FAUX	

7.4) **(0.5 point)** Un arbre complet de hauteur h=6 possède au moins 63 nœuds.

VRAI	
FAUX	X