POLYTECHNIQUE MONTRÉAL

Questionnaire examen intra

INF2010

Sigle du cours

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

Sigle et titre du cours			
INF2010 – Structures de données et algorithmes			
Professeur		*Groupe*	*Trimestre*
Ettore Merlo		Tous	Automne 2022
Jour	*Date*	*Durée*	*Heures*
☐ Lundi	17 octobre 2022	☐ 2h30 ☐	18h45 à 21h15
Documentation		*Calculatrice*	*Outils électroniques*
☒ Aucune Toute ☒ Voir directives particulières		Aucune Toutes Non-programmable (AEP)	Les appareils électroniques personnels sont interdits.

Directives particulières

- Le professeur ne répondra à aucune question durant cet examen;
- En cas de doute sur le sens d'une question, énoncez clairement toute hypothèse et supposition que vous faites et continuez;
- Un cahier d'examen vous est fourni à titre de brouillon; ne remettez pas le cahier d'examen ;
- Remettez ce questionnaire avec vos réponses dans les espaces réservés à cet effet.

Cet examen contient 5 questions sur un total de 18 pages (excluant cette page).

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

Réservé	
Q1	***/25***
Q2	***/20***
Q3	***/20***
Q4	***/15***
Q5	***/20***
Total	***/100***

Question 1: Tables de dispersement (25 points)

Soit une table de dispersement avec sondage quadratique H (clé) = (clé + i^2) % n dont l'implémentation est fournie sans modification à l'annexe 1 à titre de référence.

1.1(4 points) Insérez l'élément 84 dans le tableau suivant:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Entrées	13				95			85			88		
Active	T				T			T			T		

PHYS SIZE: 13
QUAD PROB TABLE --->
POS: 0 VAL: 13 ACT: true
POS: 4 VAL: 95 ACT: true
POS: 6 VAL: 84 ACT: true
POS: 7 VAL: 85 ACT: true
POS: 10 VAL: 88 ACT: true
<--- QUAD PROB TABLE

1.2(4 points) Insérez l'élément 53 dans le tableau suivant :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Entrées		27	14			1					40		
Active		T	T			T					T		

PHYS SIZE: 13
QUAD PROB TABLE --->
POS: 1 VAL: 27 ACT: true
POS: 2 VAL: 14 ACT: true
POS: 4 VAL: 53 ACT: true
POS: 5 VAL: 1 ACT: true
POS: 10 VAL: 40 ACT: true
<--- QUAD PROB TABLE

1.3(4 points) Partez du tableau suivant et effectuez l'opération remove (54).

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Entrées			28	15		54	2					41	
Active			T	T		T	T					T	

Donnez l'état du tableau après cet appel.

PHYS SIZE: 13
QUAD PROB TABLE --->
POS: 2 VAL: 28 ACT: true
POS: 3 VAL: 15 ACT: true
POS: 5 VAL: 54 ACT: false
POS: 6 VAL: 2 ACT: true
POS: 11 VAL: 41 ACT: true
<--- QUAD PROB TABLE

1.4(4 points) Insérez l'élément 68 dans le tableau suivant :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Entrées				29	16		55	3					42
Active				T	T		F	T					T

QUAD PROB TABLE --->
POS: 2 VAL: 68 ACT: true
POS: 3 VAL: 29 ACT: true
POS: 4 VAL: 16 ACT: true
POS: 6 VAL: 55 ACT: false
POS: 7 VAL: 3 ACT: true
POS: 12 VAL: 42 ACT: true
<--- QUAD PROB TABLE

1.5 (9 points) Soit une table de dispersement avec un NOUVEAU SONDAGE

QUADRATIQUE DIFFERENT : H (clé) = (clé + i^2 + i/2) % *n*

1.5.1 (4 points) Insérez l'élément 28 dans le tableau suivant selon ce nouveau sondage quadratique:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Entrées			15	2			19						25
Active			T	T			T						T

QUAD PROB TABLE --->
POS: 2 VAL: 15 ACT: true
POS: 3 VAL: 2 ACT: true
POS: 6 VAL: 19 ACT: true
POS: 7 VAL: 28 ACT: true
POS: 12 VAL: 25 ACT: true
<--- QUAD PROB TABLE

1.5.2 (5 points) Modifiez la routine correspondante a la signature « int findPos(AnyType x) » dans l'Annexe-1 de façon à implanter ce nouveau sondage quadratique.

```
private int findPos (AnyType x) {

    int offset = 1;
    int currentPos = myhash( x );

    int i = 0;
    while ((array[ currentPos ] != null) &&
           (!(array[ currentPos ].element.equals(x)))) {

            currentPos += offset;
            currentPos += i / 2;

            if (currentPos >= array.length)
                currentPos -= array.length;

            offset += 2;
            i++;
    }

    return currentPos;
}
```

Question 2 : Tri Fusion (Merge Sort) (20 points)

2.1 (5 points) Partez du vecteur à trier suivant :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Valeurs	47	15	99	1	30	16	53	64	27	75	3	9	86	20	69

Écrivez la séquence des appels a « mergeSort MS(L, R) tel que exécutes par l'algorithme pour trier ce vecteur. L (left) et R (right) correspondent aux valeurs des indexes reçus par un appel.

Continuez la suggestion :

0 14
0 6
0 3
0 1
0 0
1 1
2 3
2 2
3 3
4 6
4 5
4 4
5 5
6 6
7 14
7 10
7 8
7 7
8 8
9 10
9 9
10 10
11 14
11 12
11 11
12 12
13 14
13 13

Alternativement :

MS: 0 14
MS: 0 7
MS: 0 3
MS: 0 1
MS: 0 0
MS: 1 1
MS: 2 3
MS: 2 2
MS: 3 3
MS: 4 7
MS: 4 5
MS: 4 4
MS: 5 5
MS: 6 7
MS: 6 6
MS: 7 7
MS: 8 14
MS: 8 11
MS: 8 9
MS: 8 8
MS: 9 9
MS: 10 11
MS: 10 10
MS: 11 11
MS: 12 14
MS: 12 13
MS: 12 12
MS: 13 13
MS: 14 14

2.2(3 points) Donnez la complexité asymptotique de mergeSort dans le PIRE des cas.

n * log(n)

2.3(3 points) Considérez un vecteur déjà trié de façon ascendante. S'agit-il du meilleur cas pour mergeSort, tel que décrit dans le livre de Weiss, du pire des cas, du cas moyen ou aucun de ces cas? JUSTIFIEZ brièvement votre réponse.

Cas moyen, car tous les cas sont égaux pour mergeSort

2.4(3 points) Considérez un vecteur déjà trié de façon descendante. S'agit-il du meilleur cas pour mergeSort, du pire des cas, du cas moyen ou aucun de ces cas? JUSTIFIEZ brièvement votre réponse.

Cas moyen, car tous les cas sont égaux pour mergeSort

2.5 (3 points) Considérez un vecteur dans lequel tous les éléments sont égaux. S'agit-il du meilleur cas pour mergeSort, du pire des cas, du cas moyen ou aucun de ces cas? JUSTIFIEZ brièvement votre réponse.

Cas moyen, car tous les cas sont égaux pour mergeSort

2.6 (3 points) Quelles modifications ou ajouts à l'algorithme de seraient-elles nécessaires afin de trier un vecteur à l'envers avec le tri-fusion? JUSTIFIER brièvement la réponse.

Dans la routine « merge » , inverser le test de comparaison entre l'element a la position leftPos et l'element a la position rightPos

If (a[leftPos].compareTo(a[rightPos]) <= 0) {

devient

If (a[leftPos].compareTo(a[rightPos]) >= 0) {

Question 3 : Tri rapide (20points)

3.1 (6 points) Partez du vecteur à trier suivant :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Valeurs	23	41	34	97	11	0	51	93	81	17	99	76	77	64	92

Indiquez la valeur du pivot choisi pour la première récursion au départ et remplissez le tableau suivant avec les partitions gauche et droite après avoir restauré le pivot à la position "i" (plutôt que "right - 1") de Quicksort (note Quicksort est rapporté sans modifications en annexe pour votre référence).

Pivot	

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Partition gauche															
Partition droite															

```
PARTITIONS --->
    PARTITIONED ARRAY: --->
        INDEX: 0 ELEMENT: 23
        INDEX: 1 ELEMENT: 41
        INDEX: 2 ELEMENT: 34
        INDEX: 3 ELEMENT: 77
        INDEX: 4 ELEMENT: 11
        INDEX: 5 ELEMENT: 0
        INDEX: 6 ELEMENT: 51
        INDEX: 7 ELEMENT: 64
        INDEX: 8 ELEMENT: 81
        INDEX: 9 ELEMENT: 17
        INDEX: 10 ELEMENT: 76
        INDEX: 11 ELEMENT: 92
        INDEX: 12 ELEMENT: 97
        INDEX: 13 ELEMENT: 99
        INDEX: 14 ELEMENT: 93
    <--- PARTITIONED ARRAY

    INDEXES
        I: 11
        J: 10
        LEFT: 0
        PIVOT: 11
        RIGHT: 14

    PIVOT VALUE:   92

    LEFT PARTITION --->
        INDEX: 0 ELEMENT: 23
        INDEX: 1 ELEMENT: 41
        INDEX: 2 ELEMENT: 34
        INDEX: 3 ELEMENT: 77
        INDEX: 4 ELEMENT: 11
        INDEX: 5 ELEMENT: 0
        INDEX: 6 ELEMENT: 51
        INDEX: 7 ELEMENT: 64
        INDEX: 8 ELEMENT: 81
        INDEX: 9 ELEMENT: 17
        INDEX: 10 ELEMENT: 76
    <--- LEFT PARTITION
    PIVOT --->
        INDEX: 11 ELEMENT: 92
    <--- PIVOT
    RIGHT PARTITION --->
        INDEX: 12 ELEMENT: 97
        INDEX: 13 ELEMENT: 99
        INDEX: 14 ELEMENT: 93
    <--- RIGHT PARTITION
<--- PARTITIONS
```

3.2(14 points) Comparez la complexité ASYMPTOTIQUE de quickSort (sans démonstration) si on effectuait les variations suivantes concernant le choix du pivot et cochez les cases appropriées

3.2.1 (3.5 points) Le pivot est toujours choisi correspondant à

l'élément central (left + right - 1) / 2

dans chaque vecteur correspondant à une partition à trier :

	Amélioration	Dégradation	Pareil
Meilleur			X
Moyen			X
Pire			X

3.2.2(3.5 points) Le pivot est toujours choisi correspondant à une position aléatoire dans chaque vecteur correspondant à une partition à trier :

	Amélioration	Dégradation	Pareil
Meilleur			X
Moyen			X
Pire			X

3.2.3(3.5 points) Le pivot est toujours choisi correspondant à la médiane sur 5 éléments dans chaque vecteur à trier. Les quatre éléments considérés pour la médiane sont aux positions suivantes (calculées en arithmétique sur les entiers) :

v[left],

v[(left + right - 1) / 4],

v[1], v[(left + right -

1) / 2], v[3 * (left +

right - 1) / 4], v[right -

1]

	Amélioration	Dégradation	Pareil
Meilleur			X
Moyen			X
Pire			X

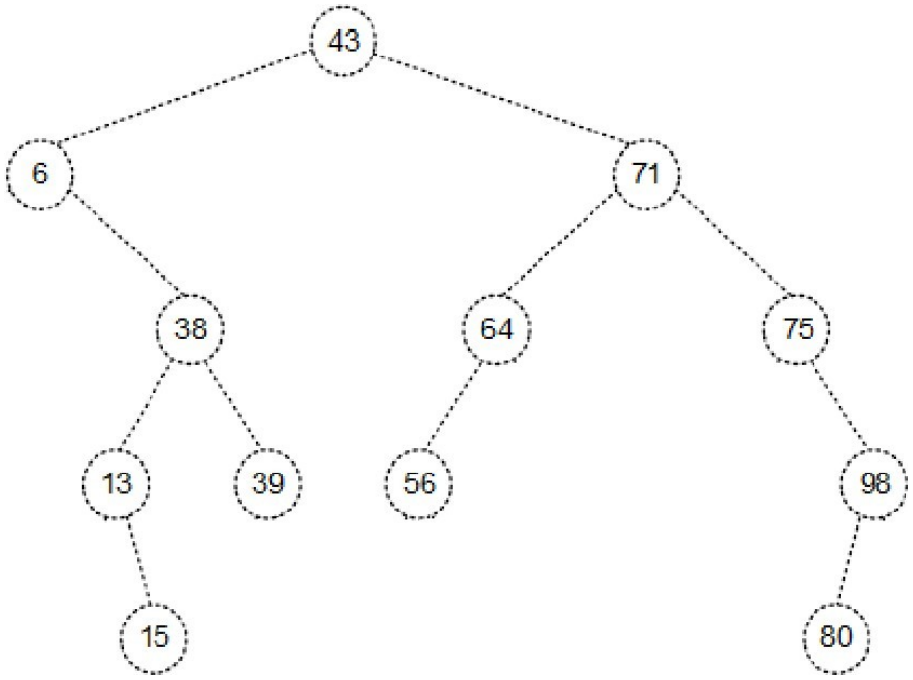
3.2.4 (3.5 points) Le pivot est toujours choisi correspondant à la médiane de tous les éléments pris dans la première moitié du vecteur à trier. Les éléments considérés pour la médiane sont aux positions suivantes (calculées en arithmétique sur les entiers) :

v[0], v[1], v[2], … v[(left + right - 1) / 2]

	Amélioration	Dégradation	Pareil
Meilleur		X	
Moyen		X	
Pire		X	

Question 4 : Parcours d'arbres (15 points)

Considérez l'arbre suivant :



4.1 (9 points) Remplissez le tableau suivant avec les parcours d'arbres indiqués.

ATTENTION : Traversez les enfants d'un nœud dans l'ORDRE DE GAUCHE À DROITE selon la figure.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
En-ordre														
Post-ordre														
Par niveau														

En-ordre : 6, 13, 15, 38, 39, 43, 56, 64, 71, 75, 80, 98
Post-ordre : 15, 13, 39, 38, 6, 56, 64, 80, 98, 75, 71, 43
Niveaux : 43, 6, 71, 38, 64, 75, 13, 39, 56, 98, 15, 80

4.2 (3 points) quelle est la complexité asymptotique du traversement d'un arbre en post-ordre avec un algorithme récursif dans le meilleur cas, le pire des cas et le cas moyen? JUSTIFIEZ brièvement vos réponses.

O(n), car on passe une seule fois sur chaque nœud pour faire un nombre d'opérations fini

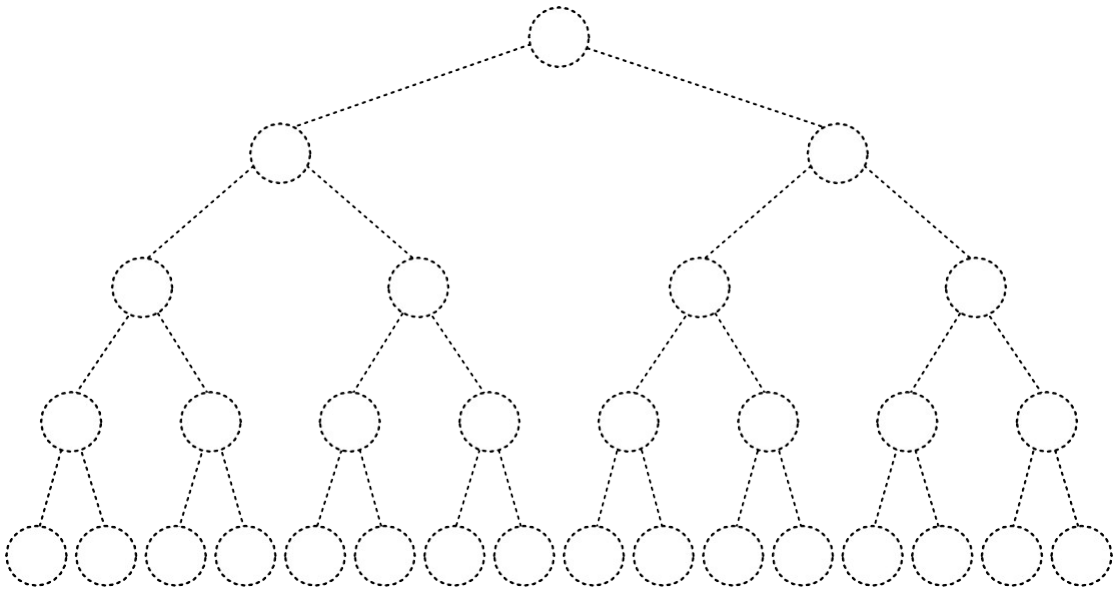
4.3(3 points) quelle est la complexité asymptotique du traversement d'un arbre par niveau en utilisant une file dans le meilleur cas, le pire des cas et le cas moyen? JUSTIFIEZ brièvement vos réponses.

O(n), car on passe une seule fois sur chaque nœud pour faire un nombre d'opérations fini

Question 5 : Arbres binaires de recherche (20 points)

5.1 (5 points) Remplissez l'arbre binaire de recherche dans la figure suivante, avec le résultat de l'insertion en séquence des entiers suivants:

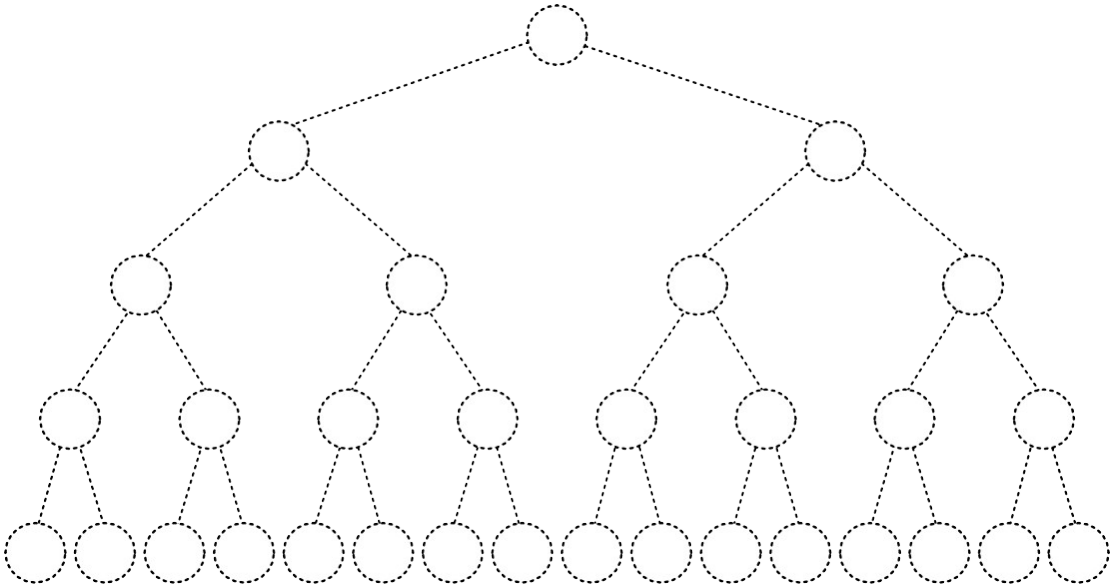
28 , 94 , 69 , 27 , 5 , 80 , 88 , 34 , 49 , 0 , 7



```
BIN TREE 1 --->

28      (1, ROOT)
  27    (2, LEFT)
    5   (3, LEFT)
      0 (4, LEFT)
      7 (4, RIGHT)
  94    (2, RIGHT)
    69  (3, LEFT)
      34        (4, LEFT)
        49      (5, RIGHT)
      80        (4, RIGHT)
        88      (5, RIGHT)

<--- BIN TREE 1
```

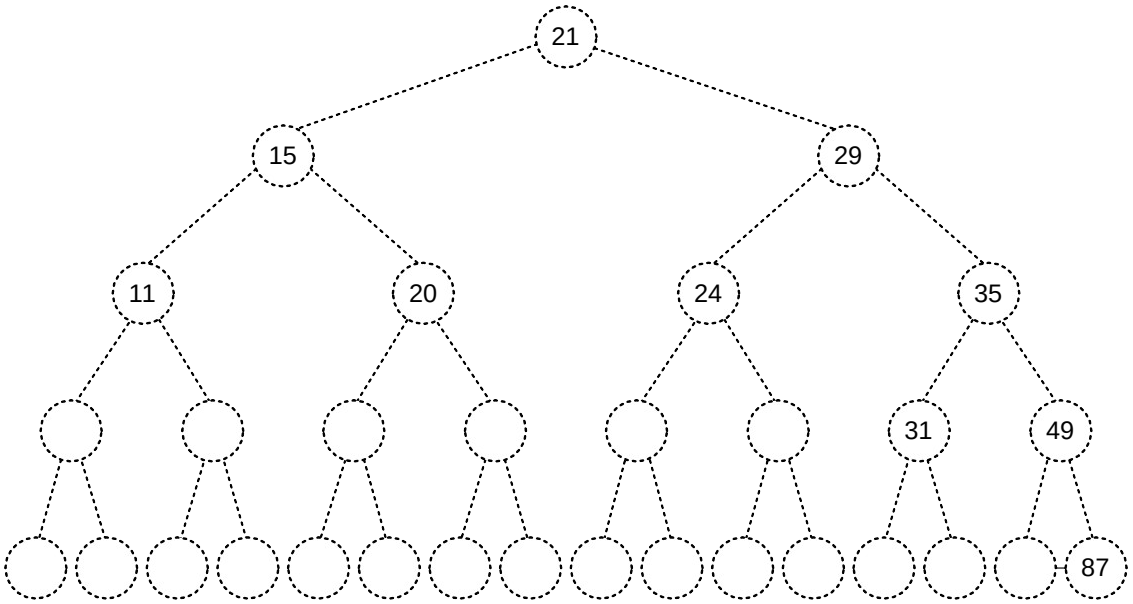
5.2(5 points) Remplissez l'arbre binaire de recherche dans la figure suivante, avec le résultat de l'insertion en séquence des entiers suivants:

5 , 1 , 47 , 83 , 40 , 56 , 35 , 55 , 24 , 39

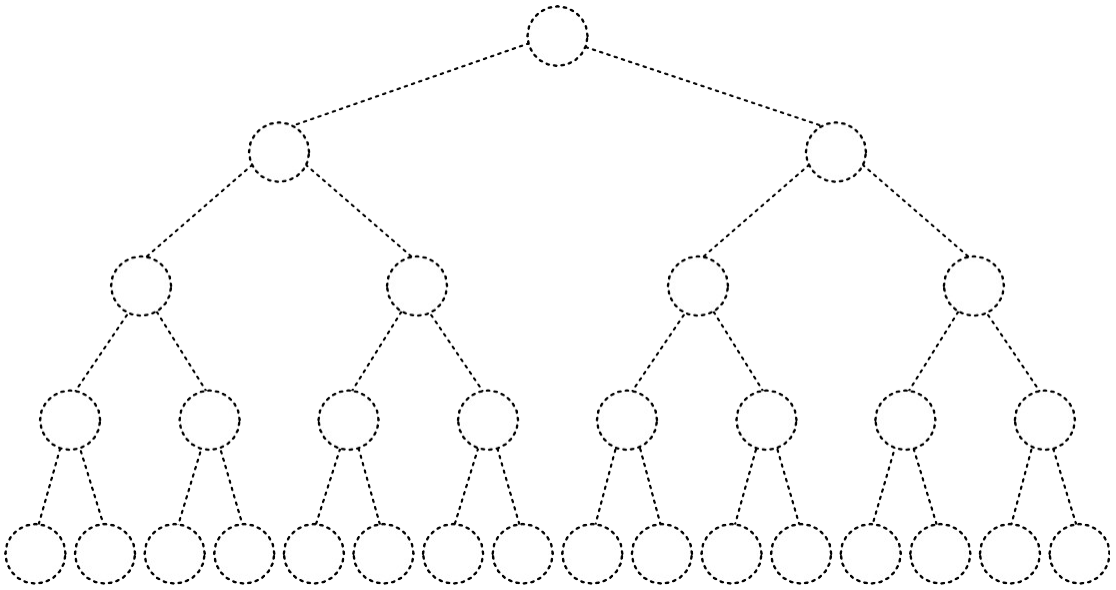


```
BIN TREE 2 --->

5       (1, ROOT)
  1     (2, LEFT)
  47    (2, RIGHT)
    40  (3, LEFT)
      35        (4, LEFT)
        24      (5, LEFT)
        39      (5, RIGHT)
    83  (3, RIGHT)
      56        (4, LEFT)
        55      (5, LEFT)

<--- BIN TREE 2
```

5.3(5 points) A partir de l’arbre en figure suivante:

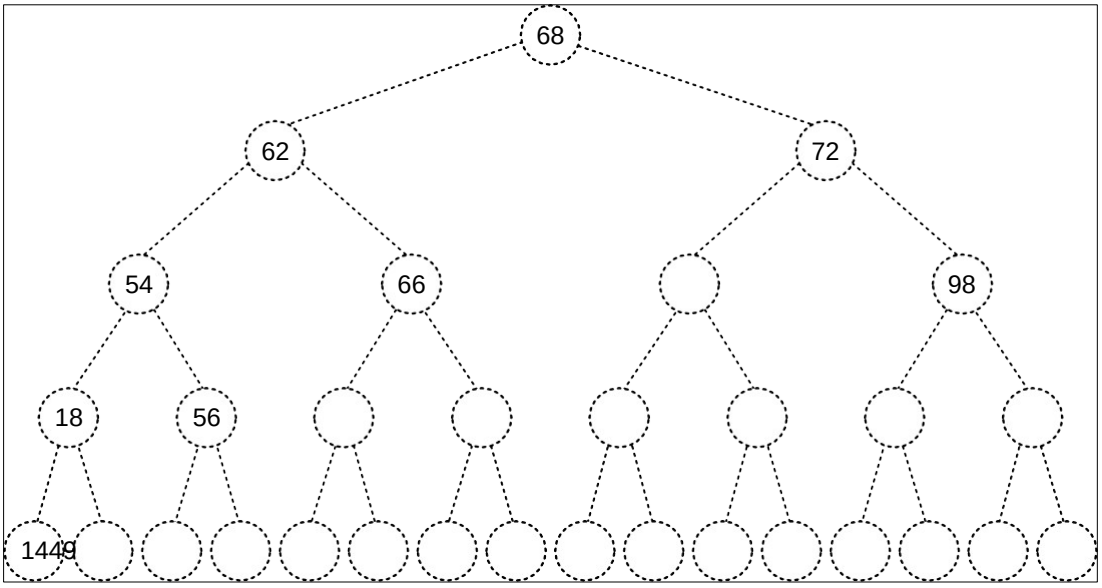


Remplissez l'arbre binaire de recherche dans la figure suivante, avec le résultat de l'élimination de l’élément 29

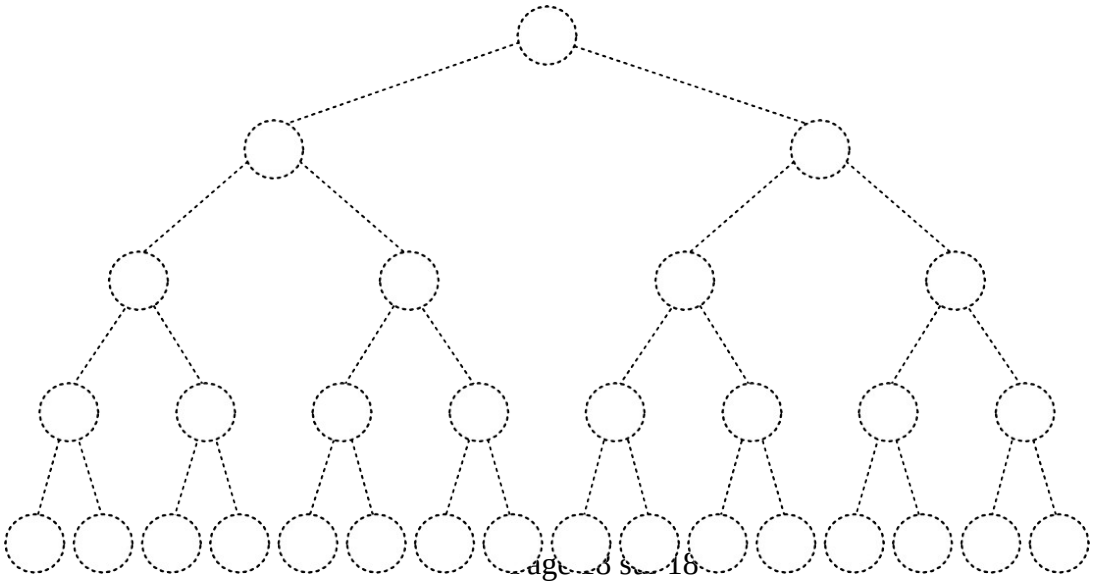


```
21      (1, ROOT)
  15    (2, LEFT)
    11  (3, LEFT)
    20  (3, RIGHT)
  31    (2, RIGHT)
    24  (3, LEFT)
    35  (3, RIGHT)
      49        (4, RIGHT)
        87      (5, RIGHT)

<--- BIN TREE 3
```

5.4(5 points) À partir de l'arbre en figure suivante,



Remplissez l'arbre binaire de recherche dans la figure suivante, avec le résultat de l'élimination de l'élément 68 :



```
72      (1, ROOT)
  62    (2, LEFT)
    54  (3, LEFT)
      18        (4, LEFT)
        14      (5, LEFT)
        49      (5, RIGHT)
      56        (4, RIGHT)
    66  (3, RIGHT)
  98    (2, RIGHT)

<--- BIN TREE 4
```