**By: Kirti JOYEJOB #6046002**

**COMP 352: DATA STRUCTURES AND ALGORITHMS**

**ASSIGNMENT 3&4**

**QUESTION 1.**

L

S

U

N

F

S

H

I

F

T

A

I

T

O

O

**QUESTION 2.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ϕ** | **L** | **S** | **U** | **H** | **S** | **F** | **N** | **T** | **I** | **I** | **A** | **T** | **F** | **ϕ** | **ϕ** |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ϕ** | **ϕ** | **ϕ** | **ϕ** | **ϕ** | **ϕ** | **ϕ** | **ϕ** | **O** | **O** | **ϕ** | **ϕ** | **ϕ** | **ϕ** | **ϕ** | **Φ** |

16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

**QUESTION 3.**

17

27

10

13

16

19

20

15

9

13

17

11

17

27

10

23

16

19

20

15

10

6

9

13

17

11

17

27

10

13

16

19

20

15

6

10

9

10

13

17

11

17

27

18

13

16

19

20

15

9

removeMin()

Output: '6'

10

10

17

13

17

11

27

18

13

16

19

20

15

removeMin()

10

Output: '9'

11

10

17

13

17

15

27

13

16

19

20

18

removeMin()

10

Output: '10'

11

13

17

16

17

15

27

23

19

20

18

**QUESTION 4.**

add: 15 add:27 add:20

15

15

15

27

20

27

add:19

15

27

20

19

add: 16

15

16

20

19

27

add: 13

13

16

15

20

19

27

add: 10

10

16

13

20

19

27

15

add: 17

10

16

13

15

20

19

17

27

add: 11

10

11

13

15

20

19

16

277

17

add:17

10

11

13

15

20

17

16

19

27

17

add:23

10

11

13

15

20

17

16

23

19

27

17

add:9

9

11

10

15

13

17

16

20

23

19

27

17

add: 10

9

11

10

15

10

17

16

13

20

23

19

27

17

add:6

6

11

9

10

10

17

16

15

13

20

23

19

27

17

add: 18

6

11

9

10

10

17

16

18

15

13

20

23

19

27

17

**QUESTION 5.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| element | 6 | 11 | 9 | 16 | 17 | 10 | 10 | 27 | 17 | 19 | 23 | 20 | 13 | 15 | 18 | Ø |

The root is at index 0 and the length of the array-list is 15.

**QUESTION 6.**

1. h(k) = k mod 11

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| h | 5 | 38 | 127 | 21 | 16 | 210 | 28 | 19 | 150 | 201 | 332 | 95 | 123 | 10 | 30 |
| h(x) | 5 | 5 | 6 | 10 | 5 | 1 | 6 | 8 | 7 | 3 | 2 | 7 | 2 | 10 | 8 |

|  |  |
| --- | --- |
| 0 | Ø |
| 1 | 210 |
| 2 |  |
| 3 | 201 |
| 4 | Ø |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 | Ø |
| 10 |  |

10

21

16

38

5

30

19

95

150

28

127

123

332

1. 7 collisions in total.

**QUESTION 7.**

h(x) = k mod 12

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| h | 5 | 38 | 127 | 21 | 16 | 210 | 28 | 19 | 150 | 201 | 332 | 95 | 123 | 10 | 30 |
| h(x) | 5 | 2 | 7 | 9 | 4 | 6 | 4 | 7 | 6 | 9 | 8 | 11 | 3 | 10 | 6 |

|  |  |
| --- | --- |
| 0 | Ø |
| 1 | Ø |
| 2 | 38 |
| 3 | 123 |
| 4 |  |
| 5 | 5 |
| 6 |  |
| 7 |  |
| 8 | 332 |
| 9 |  |
| 10 | 10 |
| 11 | 95 |

28

16

30

150

210

19

127

201

21

Number of collisions: 5  
Yes, because when the load factor is reduced, either the number of entries is reduced or the array size is increased which implies that the lesser the number of entries means lesser collisions and if the array size is increased, the chance of one element colliding with another one is lower.

**QUESTION 8.**

(i)

|  |  |  |  |
| --- | --- | --- | --- |
| h | h(x) | d(k) | Probes |
| 25 | 6 | 3 | 6 |
| 12 | 12 | 2 | 12 |
| 42 | 4 | 7 | 4 |
| 31 | 12 | 4 | 12 16 |
| 35 | 16 | 7 | 16 4 11 |
| 39 | 1 | 3 | 1 |
| 48 | 10 | 1 | 10 |
| 18 | 18 | 3 | 18 |
| 29 | 10 | 6 | 10 16 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| *key* | Ø | 39 | Ø | 29 | 42 | Ø | 25 | Ø | Ø | Ø | 48 | 35 | 12 | Ø | Ø | Ø | 31 | Ø | 18 |

1. 3 occupied clusters [indices 10, 11 and 12]
2. 5 collisions
3. Load factor = = ≈ 0.47

**QUESTION 9.**

Keys from question 8:

|  |  |
| --- | --- |
| h | h(x) = k mod 19 |
| 25 | 6 |
| 12 | 12 |
| 42 | 4 |
| 31 | 12 |
| 35 | 16 |
| 39 | 1 |
| 48 | 10 |
| 18 | 18 |
| 29 | 10 |

Keys from question 9:

|  |  |
| --- | --- |
| k | h(k) = k mod 19 |
| 29 | 10 |
| 53 | 15 |
| 14 | 14 |
| 95 | 0 |
| 32 | 13 |
| 19 | 0 |
| 30 | 11 |
| 12 | 12 |
| 72 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| *keys* | 95 | 39 | 32 | 19 | 42 | 30 | 25 | 12 | 72 | Ø | 48 | 29 | 12 | 31 | 29 | 53 | 35 | 14 | 18 |

1. 18 clusters.

Complexity is O(n) since the array should be traversed until an available space is found. The worst case is when only one location is available or unused and the next key goes into the following location next to the available one, which is taken. The whole array should, therefore, be traversed until the available one is found. Actually it is traversed O(n-1) times which is O(n).

1. 7 collisions.

**QUESTION 10.**

Yes, there are errors in the supposedly AVL tree:

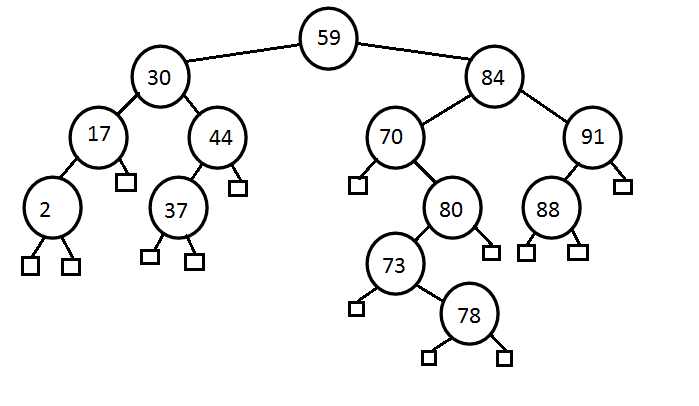
(a)Node with key 2 is not in its right position. It should have been the child of the node with key 17.

(b)Node with key 80 has a height of 2 in the left and 0 in the right.

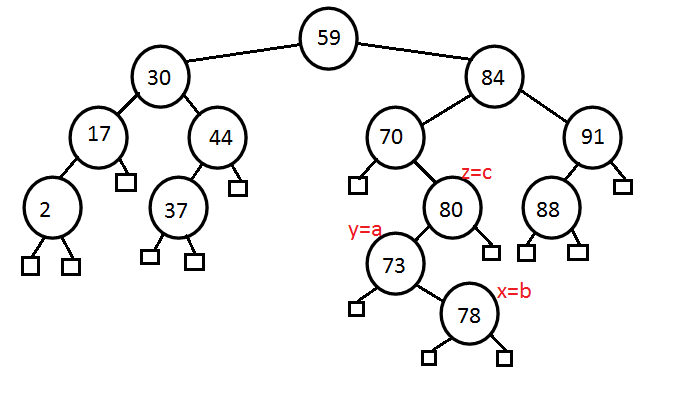
(c)Node with key 70 has a height of 3 in the right and 1 in the left.

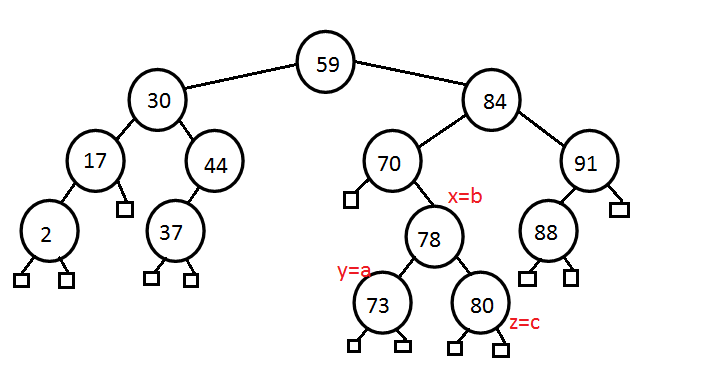
(d)Node with key 84 has a height of 4 in the left and 2 in the right.

(e)Root node with key 59 has a height of 5 in the right and 3 in the left.

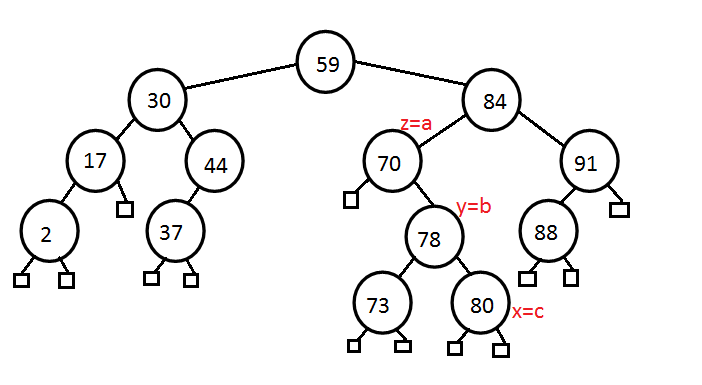
(1) Changed node with key 2 to its correct position

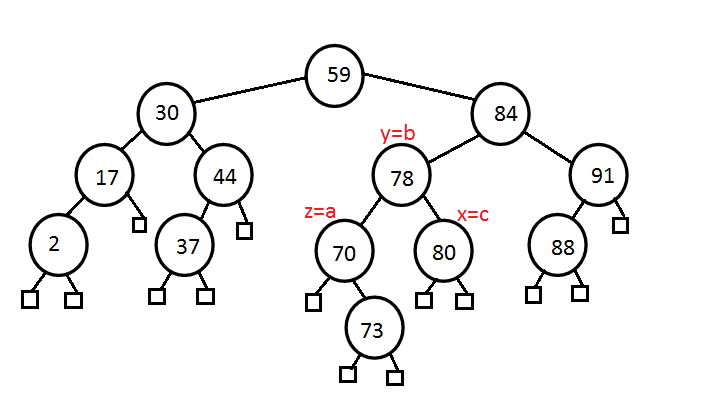
(2) Tri-node Re-structuring through node with key 80 as "z"



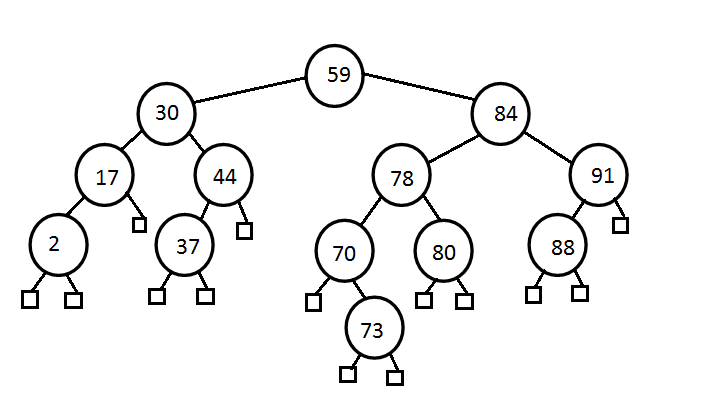


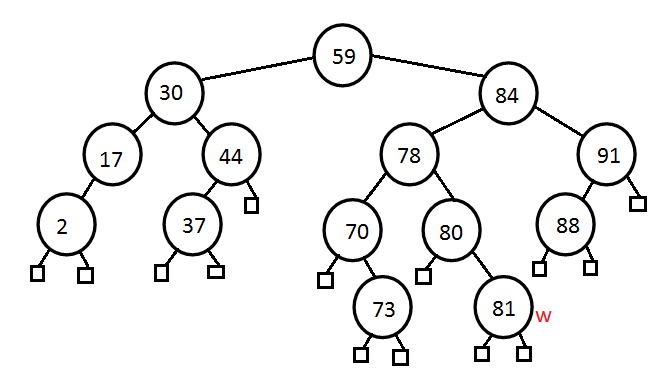
(3) Tri-node Re-structuring through node with key 70 as "z"



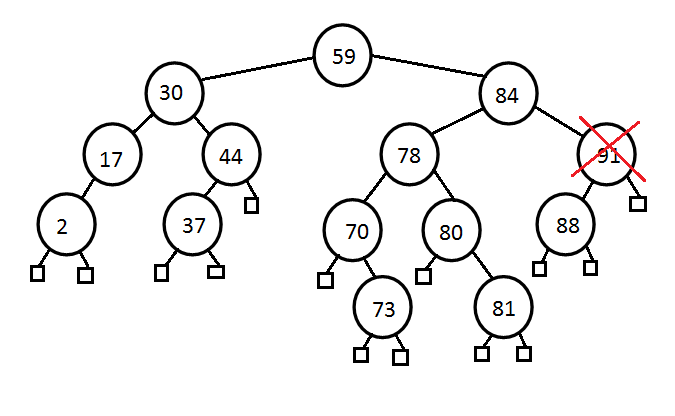


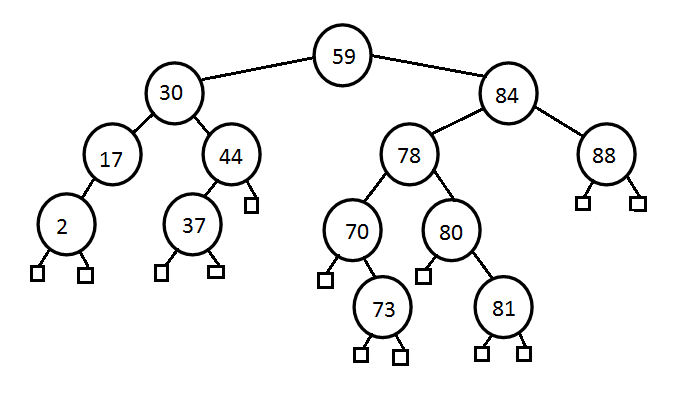
Therefore, the final AVL tree after all the necessary corrections have been made is:

(ii) put(81) which runs O(log n)

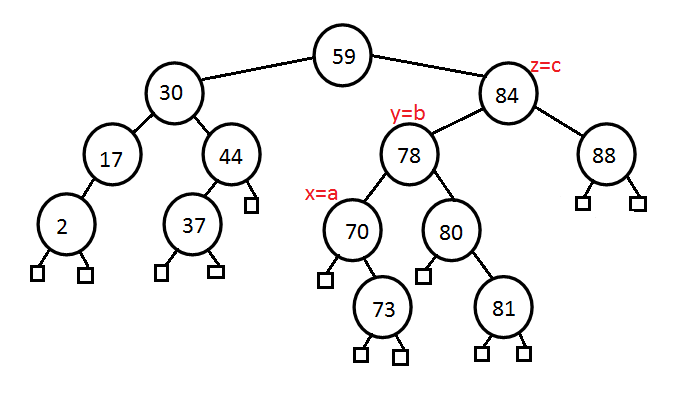


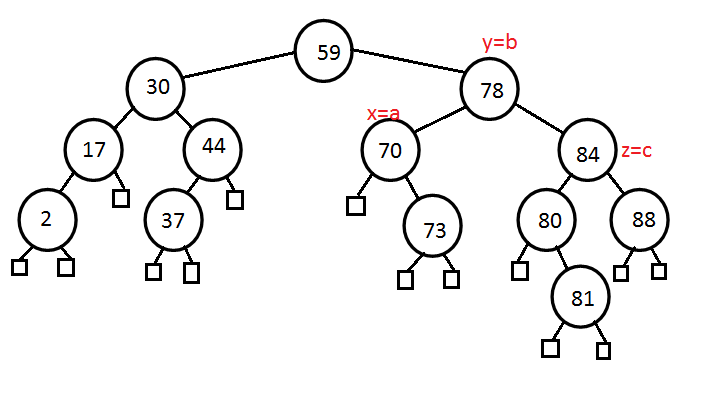
(iii) remove(91) runs in O(log n) - replace by child with key 88.



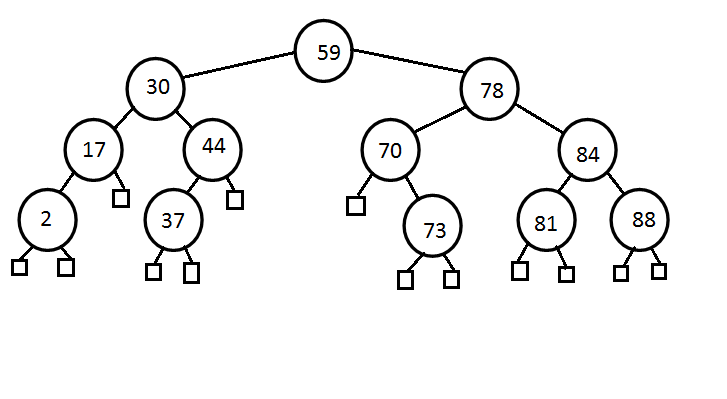


Now we have to do a rotation because operation remove(91) triggered an unbalanced AVL tree.





(iv) remove(80) runs in O(log n) - replaced node with key 80 with its children with key 81.

End of assignment