1. (a) (5 points) Illustrate the execution of the heap-sort algorithm on the following sequence: (2, 5, 16, 4, 10, 23, 39, 18, 26, 15). Show the contents of the (min) heap and the sequence at each step of the algorithm. Indicate upheap or downheap bubbling where appropriate.

Starting HEAP:

2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

4

5

18

26

15

10

23

39

16

2

Upheap

15

26

18

39

23

10

4

5

16

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Downheap

Downheap

39

18

26

16

23

10

5

4

15

26

18

10

5

39

23

16

4

15

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

39

16

23

18

26

10

4

15

5

Downheap

Downheap

39

16

23

10

18

15

26

5

39

16

23

10

18

15

5

26

Downheap

39

23

16

5

10

15

26

18

Downheap

18

39

23

16

10

5

26

15

5

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Downheap

39

16

23

18

15

10

26

26

18

23

16

15

10

39

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 |

39

23

16

10

18

26

15

Downheap

16

39

23

15

18

10

26

Downheap

Downheap

16

23

26

18

39

15

18

26

15

23

16

39

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 0 | 0 | 0 | 0 | 0 |

Downheap

39

18

26

16

23

23

16

15

26

39

18

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 16 | 0 | 0 | 0 | 0 |

Downheap

23

39

18

26

39

26

18

23

16

18

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 16 | 18 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 16 | 18 | 23 | 0 | 0 |

26

39

23

Downheap

23

26

39

39

23

26

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 16 | 18 | 23 | 26 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 16 | 18 | 23 | 26 | 39 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| min HEAP: | 2 | 4 | 5 | 10 | 15 | 16 | 18 | 23 | 26 | 39 |

Final:

39

39

26

Downheap

26

39

(b) (5 points) Illustrate the execution of the bottom-up construction of a (min) heap (like in Figure 2.49) on the following sequence: (2, 5, 16, 4, 10, 23, 39, 18, 26, 15, 7, 9, 30, 31, 40).

n=2h-1=15

Upheap

31

40

30

40

31

40

9

7

Upheap

7

9

9

7

31

30

40

15

9

7

26

15

9

31

30

31

30

Downheap

15

40

9

26

7

Downheap

15

7

9

40

26

31

30

15

26

9

18

39

18

7

40

23

39

18

10

10

UpHeap

4

4

16

10

DownHeap

16

10

4

39

18

5

23

23

39

18

10

4

16

5

18

23

39

16

4

5

10

7

10

16

40

23

39

18

31

30

31

30

5

4

5

4

2

26

9

15

26

15

9

10

16

39

2

7

18

23

40

2. (10 points) Let T be a (min) heap storing n keys. Give the pseudocode for an efficient algorithm for printing all the keys in T that are smaller than or equal to a given query key x (which is not necessarily in T). You can assume the existence of a O(1)-time print(key) function. For example, given the heap of Figure 2.41 and query key x = 7, the algorithm should report 4,5,6,7. Note that the keys do not need to be reported in sorted order. Your algorithm should run in O(k) time, where k is the number of keys reported.

Algorithm printless(T, x)

Input: a binary tree T, a key x

Output: print out all keys <= x

if(root >x)

return null

else

print(root)

if(there is a left child)

printless(the sub-tree whose root is left child, x)

if(there is a right child)

printless(the sub-tree whose root is right child, x)

3. Use the table below to convert a character key to an integer for the following questions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Letter | A | B | C | D | E | F | G | H | I | J | K | L | M |
| Key | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Letter | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| Key | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

(a) (5 points) Give the contents of the hash table that results when the following keys are inserted in

that order into an initially empty 13-item hash table: (E1, A, S1, Y, Q, U, E2, S2, T, I, O, N). Use

h(k) = k mod 13 for the hash function for the k-th letter of the alphabet (see above table for

converting letter keys to integer values). Use linear probing.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value |  |  |  |  | E |  |  |  |  |  |  |  |  |
| Key |  |  |  |  | 4 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  |  | E |  |  |  |  |  |  |  |  |
| Key | 0 |  |  |  | 4 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  |  | E | S |  |  |  |  |  |  |  |
| Key | 0 |  |  |  | 4 | 18 |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  |  | E | S |  |  |  |  |  | Y |  |
| Key | 0 |  |  |  | 4 | 18 |  |  |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S |  |  |  |  |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 |  |  |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S |  | U |  |  |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 |  | 20 |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S | E | U |  |  |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 | 4 | 20 |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S | E | U | S |  |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 | 4 | 20 | 18 |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S | E | U | S | T |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 | 4 | 20 | 18 | 19 |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S | E | U | S | T | I | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 | 4 | 20 | 18 | 19 | 8 | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A | O |  | Q | E | S | E | U | S | T | I | Y |  |
| Key | 0 | 14 |  | 16 | 4 | 18 | 4 | 20 | 18 | 19 | 8 | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A | O | N | Q | E | S | E | U | S | T | I | Y |  |
| Key | 0 | 14 | 13 | 16 | 4 | 18 | 4 | 20 | 18 | 19 | 8 | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Letter | A | B | C | D | E | F | G | H | I | J | K | L | M |
| Key | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Letter | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| Key | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

(b) (5 points) Give the contents of the hash table that results when the same keys are inserted in

that order into an initially empty 13-item hash table. Use h(k) = k mod 13 for the hash function

for the k-th letter of the alphabet (see above table for converting letter keys to integer values).

Use double hashing and let h’(k) = 1 + (k mod 11) be the secondary hash function.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value |  |  |  |  | E |  |  |  |  |  |  |  |  |
| Key |  |  |  |  | 4 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  |  | E |  |  |  |  |  |  |  |  |
| Key | 0 |  |  |  | 4 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  |  | E | S |  |  |  |  |  |  |  |
| Key | 0 |  |  |  | 4 | 18 |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  |  | E | S |  |  |  |  |  | Y |  |
| Key | 0 |  |  |  | 4 | 18 |  |  |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S |  |  |  |  |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 |  |  |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S |  | U |  |  |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 |  | 20 |  |  |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S |  | U |  | E |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 |  | 20 |  | 4 |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S |  | U | S | E |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 |  | 20 | 18 | 4 |  | 24 |  |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A |  |  | Q | E | S | T | U | S | E |  | Y |  |
| Key | 0 |  |  | 16 | 4 | 18 | 19 | 20 | 18 | 4 |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A | I |  | Q | E | S | T | U | S | E |  | Y |  |
| Key | 0 | 8 |  | 16 | 4 | 18 | 19 | 20 | 18 | 4 |  | 24 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A | I |  | Q | E | S | T | U | S | E |  | Y | O |
| Key | 0 | 8 |  | 16 | 4 | 18 | 19 | 20 | 18 | 4 |  | 24 | 14 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | A | I | N | Q | E | S | T | U | S | E |  | Y | O |
| Key | 0 | 8 | 13 | 16 | 4 | 18 | 19 | 20 | 18 | 4 |  | 24 | 14 |

4. (a) (5 points) Insert into an initially empty binary search tree items with the following keys (in this order): 30, 40, 23, 58, 48, 26, 11, 13. Draw the resulting tree after all insertions.

30

23

40

11

26

58

13

48

(b) (5 points) Remove from the binary search tree given below the following keys (in this order): 32, 65, 76, 88, 97. Draw the tree after each removal.

8088

7688

8288

6588

5488

29

28

32

97

8888

17

44

29

28

44

17

8888

97

5488

6588

8288

7688

8088

44

8888

97

5488

28

29

8288

7688

8088

17

44

8888

97

5488

28

29

8288

17

8088

97

44

5488

28

29

8288

8088

17

44

28

29

17

5488

8288

8088

(c) (5 points) A different binary search tree results when we try to insert the same sequence into an empty BST in a different order. Give an example of this with at least 5 elements and show the two different binary search trees that result.

Using: {1,2,3,4,5}

Order: {1,2,3,4,5}

5

4

3

2

1

Order: {3,4,5,1,2}

5

4

2

1

3

5. (10 points) Let T be a binary search tree, and let x be a key. Give an efficient algorithm for finding the smallest key y in T such that y > x. Note that x may or may not be in T. Explain why your algorithm has the running time it does.

Algorithm findMin>(T, x):

Input: a tree T, a key x

Output: the smallest key in the tree whose value is greater than x

y<- 10^10

if(root<y && root>x)

y<- root

if(root has a left child && left child<y) findMin>(sub-tree whose root is left child, x)

if(root has a right child && right child <y) findMin>(sub-tree whose root is right child, x)

The above algorithm should have a worst case running time of O(n) in which case all keys have values less than x, and the entire tree must be checked. This algorithm will be invoked on every node whose value is less than the current y value and if said node is less than the current value of y and greater than the passed value of x the key of said node will be the new value of y.