Name: CHAN King Yeung

SID: 1155119394

CSCI2100 Assignment 1

Question 1

1. The first Pop() returns 3

The second Pop() returns 5

The third Pop() returns 7

1. The first Dequeue() returns 9

The second Dequeue() returns 6

The third Dequeue() returns 3

1. The final result is

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | |  | | Next is  1  Push(1) | |  | | --- | |  | |  | |  | |  | | 1 | | Next is  2  Push(2) | |  | | --- | |  | |  | |  | | 2 | | 1 | | Next is  +  Pop()  Push(3) | |  | | --- | |  | |  | |  | |  | | 3 | | Next is  7  Push(7) | |  | | --- | |  | |  | |  | | 7 | | 3 | | Next is  8  Push(8) | |  | | --- | |  | |  | | 8 | | 7 | | 3 | | Next is  4  Push(4) | |  | | --- | |  | | 4 | | 8 | | 7 | | 3 | | Next is  /  Pop()  Push(2) |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| |  | | --- | |  | |  | | 2 | | 7 | | 3 | | Next is  -  Pop()  Push(5) | |  | | --- | |  | |  | |  | | 5 | | 3 | | Next is  \*  Pop()  Push(15) | |  | | --- | |  | |  | |  | |  | | 15 | | Next is  5  Push(5) | |  | | --- | |  | |  | |  | | 5 | | 15 | | Next is  6  Push(6) | |  | | --- | |  | |  | | 6 | | 5 | | 15 | | Next is  -  Pop()  Push(-1) | |  | | --- | |  | |  | |  | | -1 | | 15 | | Next is  \*  Pop()  Push(-15) | |  | | --- | |  | |  | |  | |  | | -15 | |

1. The symbol list is balanced

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | | Next is  {  Push({) | |  | | --- | |  | |  | |  | | { | | Next is  (  Push(() | |  | | --- | |  | |  | | ( | | { | | Next is  )  Pop() | |  | | --- | |  | |  | |  | | { | | Next is  {  Push({) | |  | | --- | |  | |  | | { | | { | | Next is  [  Push([) | |  | | --- | |  | | [ | | { | | { | | Next is  ]  Pop() | |  | | --- | |  | |  | | { | | { | | Next is  }  Pop() |
|  |  |  |
| |  | | --- | |  | |  | |  | | { | | Next is  }  Pop() | |  | | --- | |  | |  | |  | |  | |

Question 2

1. It is a max tree but not a complete binary tree. Thus, it is not a max heap

A close up of text on a black background

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 50 | 40 | 45 | 35 | 33 | 20 |  |  |  |  |  |  | 15 | 6 |
| [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |

|  |  |  |  |
| --- | --- | --- | --- |
| A close up of text on a black background  Description automatically generated | The next position available is 11 | A close up of text on a black background  Description automatically generated | If we put 45 at position 11, it violated the property of max heap  Interchange node 45 and node 25 |
| A close up of text on a black background  Description automatically generated | If we put 45 at position 5, it violated the property of max heap  Interchange node 45 and node 37 | A close up of text on a black background  Description automatically generated | If we put 45 at position 2, it does not violate the property of max heap  The insertion is done |

|  |  |  |  |
| --- | --- | --- | --- |
| A close up of text on a black background  Description automatically generated | Delete the root of max heap | A close up of a mans face  Description automatically generated | Replace the root by node 18 |
| A close up of text on a black background  Description automatically generated | To maintain the max heap property, swap node 18 and node 40 | A close up of text on a black background  Description automatically generated | To maintain the max heap property, swap node 18 and node 30 |
| A close up of text on a black background  Description automatically generated | There is no violation of max heap property  The deletion is done |  |  |

Question 3

A close up of a map

Description automatically generated

1. Node 30 is the successor of node 29
2. Node 40 is the predecessor of node 42

|  |  |  |
| --- | --- | --- |
| Delete node 50  A close up of text on a black background  Description automatically generated | Delete node 10  A close up of text on a black background  Description automatically generated | Delete node 20  A close up of a map  Description automatically generated |

Question 4

1. max(root)

node = root

while !isEmpty(node) and !isEmpty(rightChild(node))

node = rightChild(node)

return node

1. isBalanced(root)

if isEmpty(root)

return 1

else if abs(height(leftChild(root)) - height(rightChild(root))) <= 1 and isBalanced(leftChild(root)) and isBalanced(rightChild(root))

return 1

else

return 0

1. kthLargestKey(root, k)

if k = rightSize(root) + 1

return data(root)

else if k <= rightSize(root)

kthLargestKey(rightChild(root), k)

else

kthLargestKey(leftChild(root), k - 1 - rightSize(root))