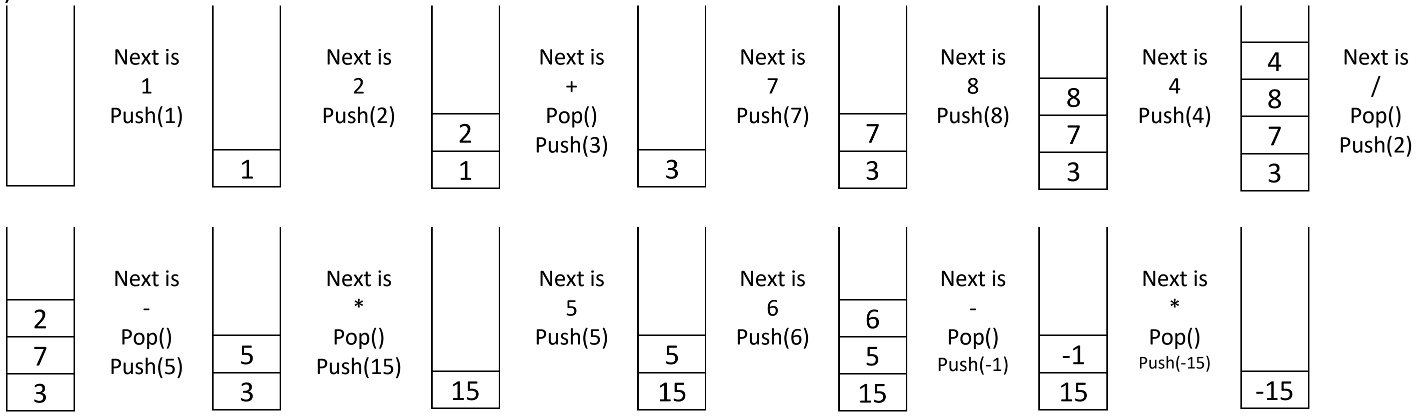
Name: CHAN King Yeung
SID: 1155119394
CSCI2100 Assignment 1

Question 1

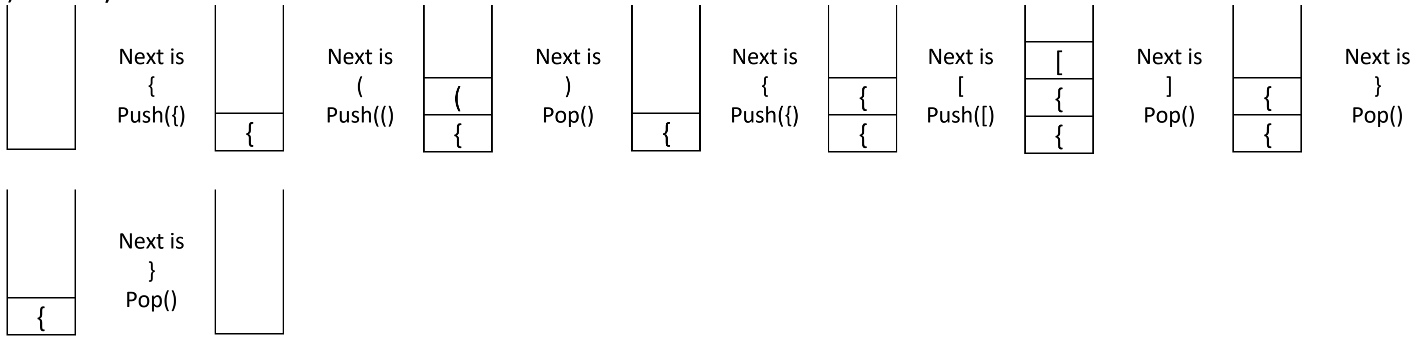
a) The first Pop() returns 3
   The second Pop() returns 5
   The third Pop() returns 7

b) The first Dequeue() returns 9
   The second Dequeue() returns 6
   The third Dequeue() returns 3

c) The final result is -15

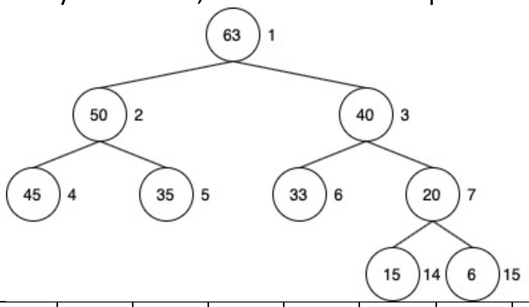


d) The symbol list is balanced



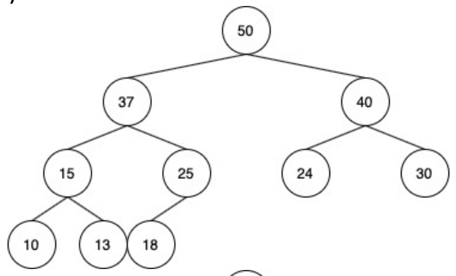
Question 2

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

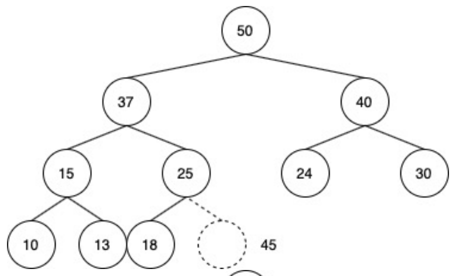


63	50	40	45	35	33	20							15	6
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]

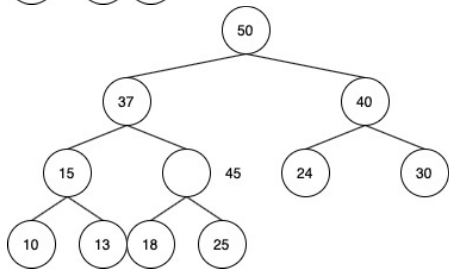
b)



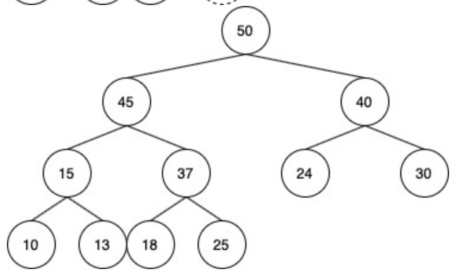
The next position available is 11



If we put 45 at position 11, it violated the property of max heap
Interchange node 45 and node 25

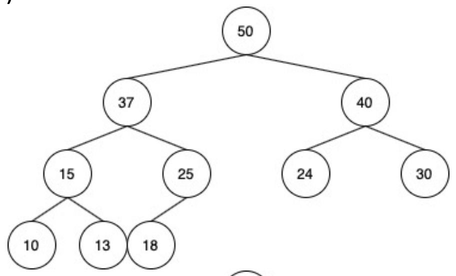


If we put 45 at position 5, it violated the property of max heap
Interchange node 45 and node 37

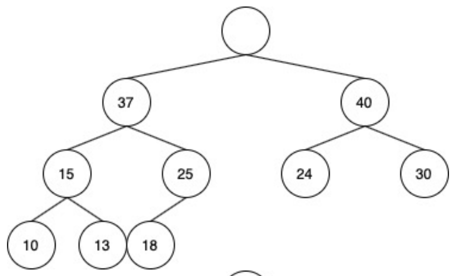


If we put 45 at position 2, it does not violate the property of max heap
The insertion is done

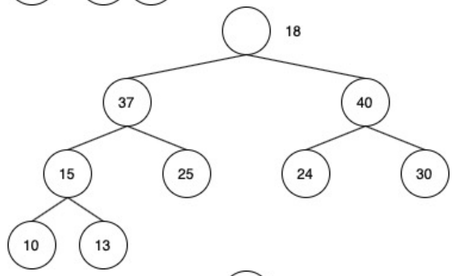
c)



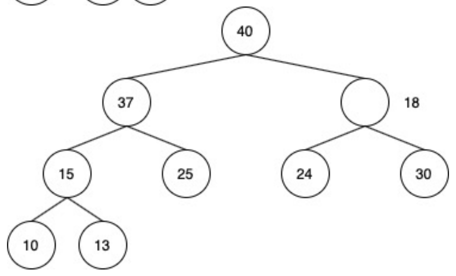
Delete the root of max heap



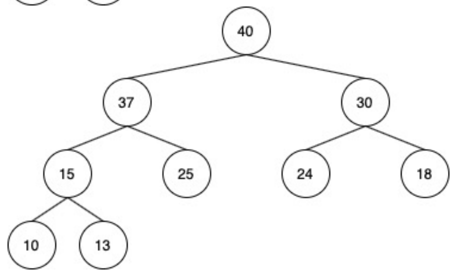
Replace the root by node 18



To maintain the max heap property, swap node 18 and node 40



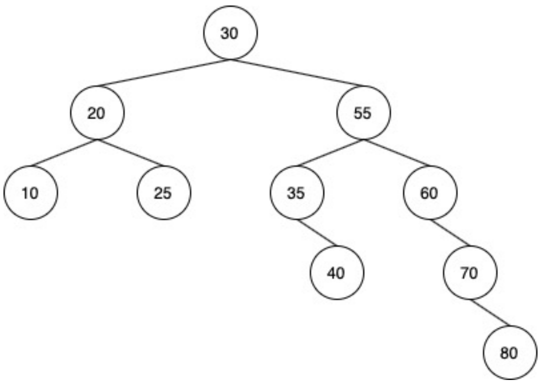
To maintain the max heap property, swap node 18 and node 30



There is no violation of max heap property
The deletion is done

Question 3

a)

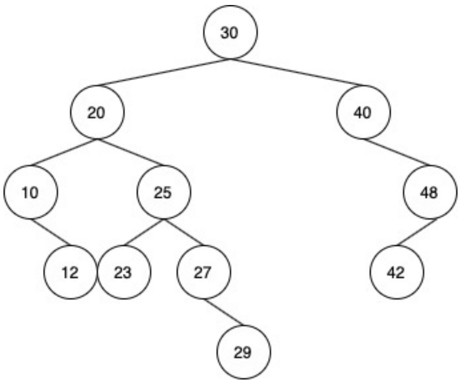


b) Node 30 is the successor of node 29

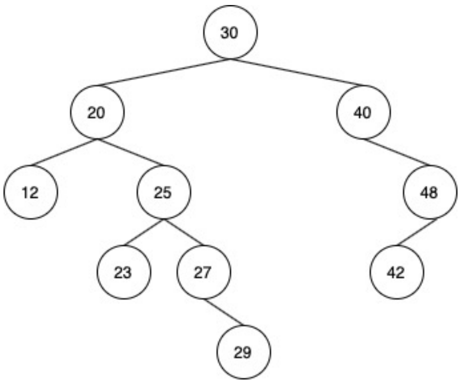
c) Node 40 is the predecessor of node 42

d)

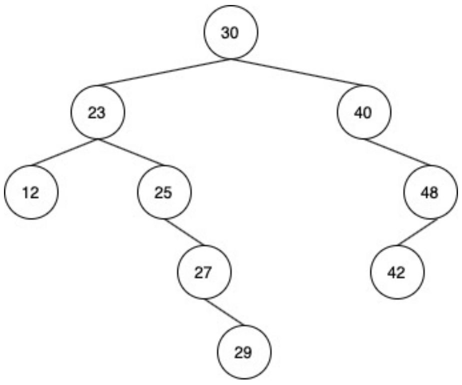
Delete node 50



Delete node 10



Delete node 20



Question 4

a)
```
max(root)
node = root
while !isEmpty(node) and !isEmpty(rightChild(node))
    node = rightChild(node)
return node
```

b)
```
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
```

c)
```
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))
```