Solution for the Midterm

Note that some answers are not stated formally; some of those are designed to help you to understand the concept better.

Quick Answer Key

1. **5**, 2. **4**, 3. **2**, 4. **2**, 5. **3**, 6. **2**, 7. **2**, 8. **4**, 9. **1**, 10. **3**, 11. **4**, 12. **2**, 13. **2**, 14. **2**

Choices for answering Questions 1, 2, 3

1. H E F A B C
2. H E F A C B
3. H E F C B A
4. I H G F E D
5. A B C E F H

Question 1. (5 points)

The following operations are carried out on an initially empty stack (in that order):

`push A, push B, push C, push D, pop, push E, push F, push G, pop, push H, push I, pop`

After the above sequence of operations, the stack contents look like (choose one, assuming the top of the stack is rightmost).

Solution. The answer is **5**. The sequence for the stack contents is:

- A
- A B
- A B C
- A B C D
- A B C
- A B C E
- A B C E F
- A B C E F G
- A B C E F
- A B C E F H
- A B C E F H G
- A B C E F H

One easy way to find the answer is that we can observe that the number of `push` calls is always greater than that of `pop` in every procedure. It implies that the stack sontents should start with 'A' which was initially pushed to the stack. ■

Question 2. (5 points)

The following operations are carried out on an initially empty stack (in that order):

`enqueue A, enqueue B, enqueue C, enqueue D, dequeue, enqueue E, enqueue F, enqueue G, dequeue, enqueue H, enqueue I, dequeue`

After the above sequence of operations, the queue contents look like (choose one, assuming the first-in is rightmost).

Solution. The answer is **4**. The sequence for the queue contents is:

- A
- B A
- C B A
- D C B A
- D C B
- E D C B
- F E D C B
- G F E D C B
- G F E D C
- H G F E D C
- I H G F E D C
- I H G F E D

One easy way to find the answer is that since we called `dequeue` three times, there should not be 'C' in the queue contents. Observe that 1,2,3 and 5 contain 'C' in the list. ■

Question 3. (5 points)

The following operations are carried out on an initially empty binary heap, using an array representation as covered in class.

`insert A, insert B, insert C, insert D, remove max, insert E, insert F, insert G, remove max, insert H, insert I, remove max`

After the above sequence of operations, the array contents look like (choose one).

Solution. The answer is **2**. The sequence for the array contents is:

- A
- A B $\xrightarrow{\text{swim up}}$ B A
- B A C $\xrightarrow{\text{swim up}}$ C A B
- C A B D $\xrightarrow{\text{swim up}}$ D C B A
- A C B $\xrightarrow{\text{sink down}}$ C A B
- C A B E $\xrightarrow{\text{swim up}}$ E C B A
- E C B A F $\xrightarrow{\text{swim up}}$ F E B A C
- F E B A C G $\xrightarrow{\text{swim up}}$ G E F A C B
- B E F A C $\xrightarrow{\text{sink down}}$ F E B A C
- F E B A C H $\xrightarrow{\text{swim up}}$ H E F A C B
- H E F A C B I $\xrightarrow{\text{swim up}}$ I E H A C B F
- F E H A C B $\xrightarrow{\text{sink down}}$ H E F A C B ■

Question 4. (5 points)

In the dynamic-array implementation of a stack, re-sizing takes place either (i) when the array becomes full, in which case it results in an array whose size is c times the size of the old one; or (ii) when the array becomes only one-quarter full, in which case it results in an array whose size is d times the size of the old one. Choose the best of the following:

1. $c = 2$ and $d = 1/4$
2. $c = 2$ and $d = 1/2$
3. $c = 2$ and $d = 3/4$
4. $c = 3/2$ and $d = 1/4$
5. $c = 3/2$ and $d = 1/2$

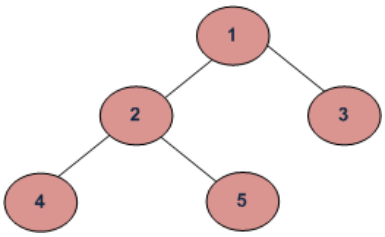
Solution. The answer is **2**. In `push()`, after checking the array size, if there is no room, we *double* the size of the array. Then we simply insert the new item. Similarly, in `pop()`, we begin by deleting the item, then we *halve* the array size if it is too large - when the stack size is less than one-fourth the array size. ■

Question 5. (5 points)

In a binary tree T (not necessarily complete), the pre-order number of node v is smaller than the pre-order number of node w, and the post-order number of v is smaller than the post-order number of w. Choose the best of the following statements.

1. v is ancestor of w in T
2. w is ancestor of v in T
3. There is a node x in T such that v is in the subtree of the left child of x, and w is in the subtree of the right child of x
4. There is a node x in T such that w is in the subtree of the left child of x, and v is in the subtree of the right child of x
5. None of the above

Solution. The answer is **3**. Let $PreOrder(x)$ denotes the pre-order number of a node x in T, and $PostOrder(x)$ denotes the post-order number of a node x in T. Note that in pre-order, tree is traversed in the order (root, left, right), and in post-order, (left, right, root). For example, in the tree below,



Preorder (Root, Left, Right) : 1 2 4 5 3, and
Postorder (Left, Right, Root) : 4 5 2 3 1.
Then we have the following case study:

- (Case 1) $PreOrder(v) < PreOrder(w)$ and $PostOrder(v) > PostOrder(w)$
- (Case 2) $PreOrder(v) > PreOrder(w)$ and $PostOrder(v) < PostOrder(w)$
- (Case 3) $PreOrder(v) < PreOrder(w)$ and $PostOrder(v) < PostOrder(w)$

- (Case 4) $PreOrder(v) > PreOrder(w)$ and $PostOrder(v) > PostOrder(w)$

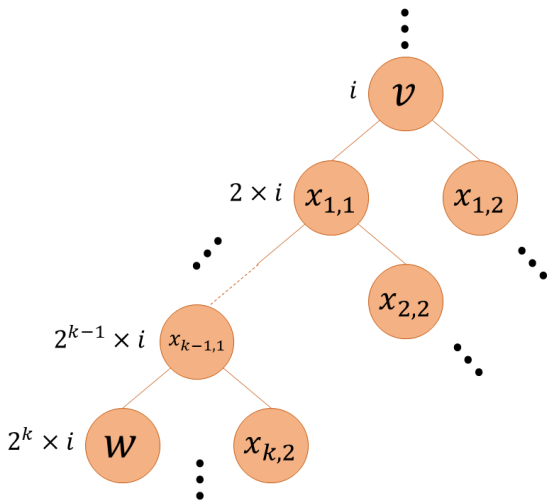
Therefore, Case 3 satisfies our condition. ■

Question 6. (5 points)

An array is used to represent a complete binary tree T. Assume node v of T is at position i in the array, and node w is at position j in the array where $j = 2^k * i$ for some positive integer k. Choose the best of the following statements, where $PreOrder(x)$ denotes the pre-order number of a node x in T, $PostOrder(x)$ denotes the post-order number of x in T, and $InOrder(x)$ denotes the in-order number of x in T,

1. $PreOrder(w) = PreOrder(v) + 2^k$
2. $PreOrder(w) = PreOrder(v) + k$
3. $PreOrder(v) = PreOrder(w) + 2^k$
4. $PreOrder(v) = PreOrder(w) + k$
5. None of the above

Solution. The answer is **2**. Since $j = 2^k * i$, we have the tree structure as below:



If we let $v = x_{0,1}$ and $w = x_{k,1}$, then we have

$$PreOrder(x_{i,1}) = PreOrder(x_{i-1,1}) + 1$$

for $i = 1, \cdots, k$. Adding all k equations together, we get $PreOrder(w) = PreOrder(x_{k,1}) = PreOrder(x_{0,1}) + k = PreOrder(v) + k$. ■

Choices for answering Questions 7 and 8.

1. causes the tree's height to increase by 1 and the number of leaves to increase by 1
2. causes the tree's height to increase by 1 and does not change the number of leaves
3. causes the tree's height to decrease by 1 and the number of leaves to decrease by 1
4. causes the tree's height to decrease by 1 and does not change the number of leaves
5. does not change the tree's height and does not change the number of leaves

The two questions that follow are independent each other (both pertain to Figure 0).

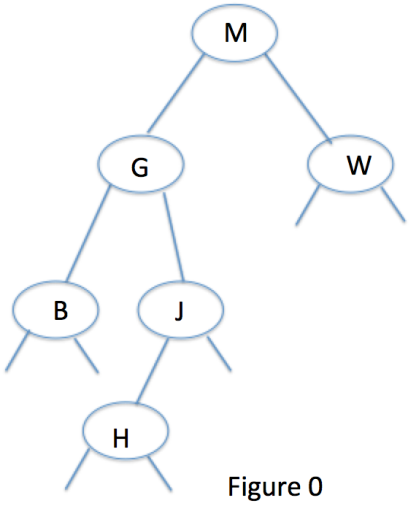


Figure 0

Question 7. (5 points)

Inserting I in the binary search tree shown in Figure 0 (choose the best answer)

Solution. The answer is **2**. We need to search `I` in the BST first.

- `I` is less than `M` so look to the left
- `I` is greater than `G` so look to the right
- `I` is less than `J` so look to the left
- `I` is greater than `H` so look to the right
- link is null

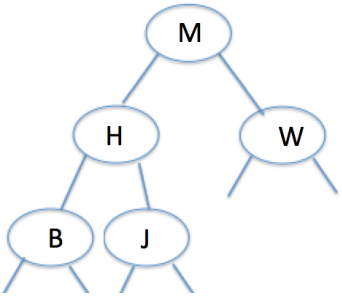
Therefore, we need to insert `I` to the right child of `H` which causes the tree's height to increase by 1 and does not change the number of leaves. ■

Question 8. (5 points)

Deleting G from the binary search tree shown in Figure 0 (choose the best answer)

Solution. The answer is **4**. We need to search `G` in the BST first. Now proceed as following:

- From the key `G`, go right, then go left until reaching null left link (left child of `H`)
- Therefore the successor is `H`
- Delete `G` by replacing it with its successor
- Update links



Therefore, deleting `G` causes the tree's height to decrease by 1 and does not change the number of leaves. ■

Choices for answering Questions 9, 10, 11.

1. one 2-node and three 3-nodes
2. three 2-nodes and one 3-node
3. seven 2-nodes and no 3-nodes
4. six 2-nodes and one 3-node
5. four 2-nodes and two 3-nodes

The next 3 questions are about the 2-3 tree that would result from inserting in the 2-3 tree of Figure 1 (the 3 questions are independent of each other, i.e., for each of them the initial 2-3 tree is the one shown on Figure 1).

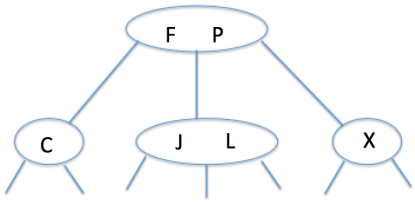


Figure 1

Question 9. (5 points)

If E is inserted in the 2-3 tree shown in Figure 1, then after re-balancing the resulting 2-3 tree has (choose the best answer)

Solution. The answer is **1**. We need to search `E` first.

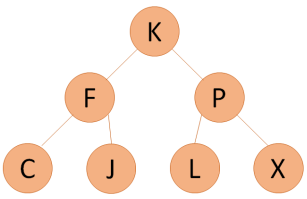
- E is less than F and P so look to the left
- E is greater than C so look to the right
- link is null
- Insert E, replacing 2-node with new 3-node containing E

Therefore, it causes deleting one 2-node and creating one 3-node. So resulting 2-3 tree has one 2-node and three 3-nodes. ■

Question 10. (5 points)

If K is inserted in the 2-3 tree shown in Figure 1, then after re-balancing the resulting 2-3 tree has (choose the best answer)

Solution. The answer is **3**. Searching K ends at the 3-node (J,L), so add new key K to 3-node to make temporary 4-node (J,K,L). Then we have to split 4-node into two 2-nodes and pass middle key K to parent. Then the parent becomes another temporary 4-node (F,K,P), so split 4-node into two 2-nodes and pass middle key K to parent. Therefore, the resulting 2-3 tree looks like a complete binary search tree which has seven 2-nodes and no 3-nodes. ■



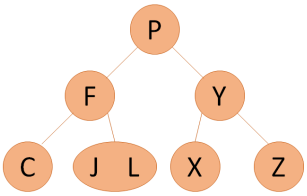
Question 11. (5 points)

If, in the 2-3 tree shown in Figure 1, we do an insert of Y followed by an insert of Z, then after re-balancing the resulting 2-3 tree has (choose the best answer)

Solution. The answer is **4**. Follow the exactly same rule as the previous answer.

- Searching Y ends at the 2-node X, so insert Y and replace 2-node with new 3-node.
- Searching Z ends at the 3-node (X,Y), so add new key Z to 3-node to make temporary 4-node (X,Y,Z). Then we have to split 4-node twice as we did in the previous question.

Therefore, the resulting 2-3 tree becomes as the figure below which has six 2-nodes and one 3-node. ■



Material for Questions 12, 13.

The red-black tree shown in Figure 2 has three red links (drawn heavier than the black links): One between T and G, one between L and J, and one between Z and X.

Choices for answering Questions 12, 13.

1. no red links
2. only one red link
3. two red links
4. three red links
5. four red links

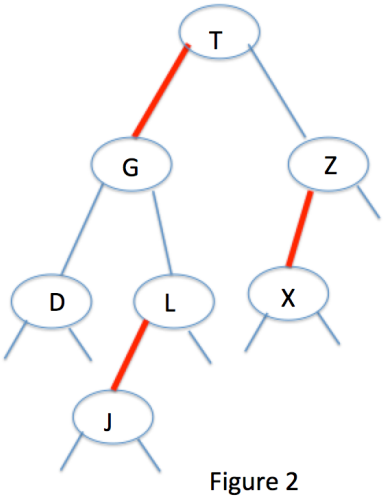
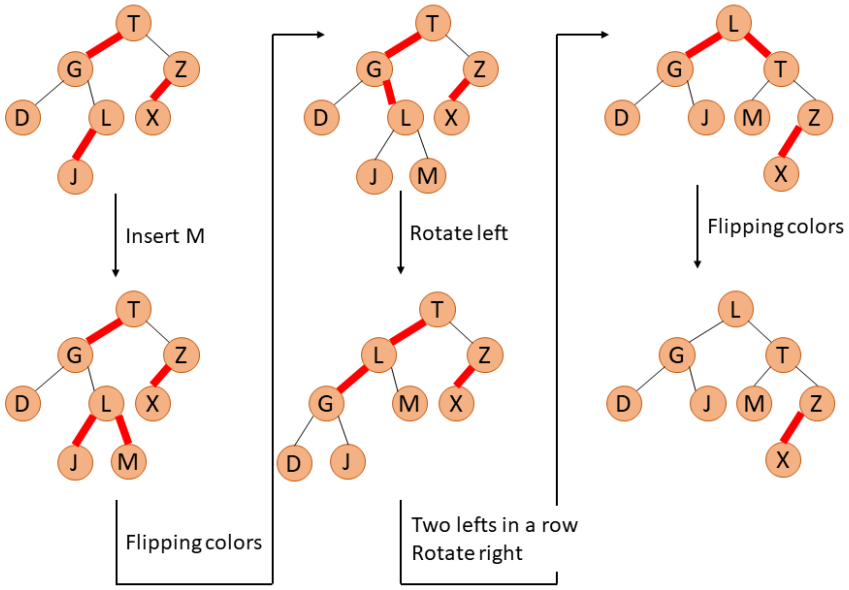


Figure 2

Question 12. (5 points)

If M is inserted in the red-black tree shown in Figure 2, then after re-balancing the resulting red-black tree has (choose the best answer)

Solution. The answer is **2**. Basically, the logic is completely same as that of 2-3 tree because there is a 1-1 correspondence between red-black BSTs and 2-3 trees.

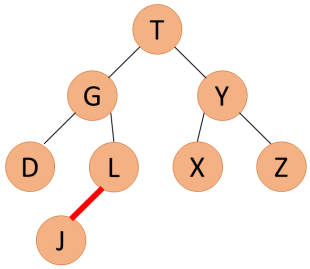


Therefore, the resulting red-black tree has only one red link. ■

Question 13. (5 points)

If Y is inserted in the red-black tree shown in Figure 2, then after re-balancing the resulting red-black tree has (choose the best answer)

Solution. The answer is **2**. Following the same logic as before, we have the following tree:



Therefore, the resulting red-black tree has only one red link. ■

Question 14. (5 points)

Suppose you have an initially empty hash table of size 7 (hence its indices are from 0 to 6), in which the keys A, B, C, D, E, F, G are inserted (in that order) using linear probing, with a hash function h such that $h(A) = 4, h(B) = 5, h(C) = 0, h(D) = 6, h(E) = 4, h(F) = 6, h(G) = 5$. Then the resulting table's contents are (choose the best, assuming leftmost has index 0 and rightmost has index 6).

1. C G F E A B D
2. C E F G A B D
3. C F G E A B D
4. C E G F A B D
5. B A G F E C D

Solution. The answer is **2**. Using linear probing, when we hash to a table index that is already occupied, then we just check the next entry in the table by incrementing the index until finding an empty table entry. Therefore, we get the following result.

key	hash	0	1	2	3	4	5	6
A	4					A		
B	5					A	B	
C	0	C				A	B	
D	6	C				A	B	D
E	4	C	E			A	B	D
F	6	C	E	F		A	B	D
G	5	C	E	F	G	A	B	D

Therefore, the resulting table's contents are C E F G A B D. ■