EECS 560 Midterm Exam Section 2

Note: This exam is closed-book and closed-notes

Name: **Chase Odgers** **KU ID:** **2868660**

(1) Given a real-world application of post order traversal, explain why post order is needed for this application. (10pts)

The benefits of post order traversal can be seen when needing to destroy a tree. This is because the last visited node is the root node, allowing for all child nodes to be deleted before their parent nodes.

Another benefit of the post order traversal can also be seen when needing to obtain the postfix expression of a given problem like in problem 2 of this exam. (I resolved problem 2 using a stack example as that is what we did in our labs however it could also be done as an expression tree gone through in post order)

(2) Convert the infix expression 128 + (7 * (12 - 3)) + 12 into a postfix expression. Show all steps. (10pts)

Input	**Stack**	**Print**
128		**128**
+	+	**128**
(	**+(**	**128**
7	**+(**	**128 7**
*****	**+(***	**128 7**
(	**+(*(**	**128 7**
12	**+(*(**	**128 7 12**
-	**+(*(-**	**128 7 12**
3	**+(*(-**	**128 7 12 3**
)	**+(***	**128 7 12 3 -**
)	+	**128 7 12 3 - ***
+	+	**128 7 12 3 - * +**
12	+	**128 7 12 3 - * + 12**
		128 7 12 3 - * + 12 +

(3) Calculate the final result of the postfix expression from the previous question. Show all steps. (10pts)

Input	**Stack**	**Calculate and push to stack**
128	**128**	
7	**128 7**	
12	**128 7 12**	
3	**128 7 12 3**	
-	**128 7**	**12 - 3**
*****	**128 7 9**	
+	**128**	**9 * 7**
12	**128 63**	**63 + 128**
+	**191 12**	
	empty stack	**203**
Result now at top of stack	**203**	

The result of 128 + (7 * (12 - 3)) + 12 **is <u>203.</u>**

(4) What is the advantage of AVL tree over binary search tree? What is the advantage of B-tree over binary search tree? (10pts)

AVL: The advantage of an AVL tree over a standard BST is that an AVL tree balances. The worst case time complexity of a standard BST is O(n) which happens as a result of an ordered list being fed in, so for a list *n* units long the tree from the head node continues to place the next *n* node to the right of the *n-1* node resulting in a tree of all right children and no left children. By balancing the tree you avoid this problem by not allowing the head nodes left tree to have a height difference of more than 1 than the right tree, ensuring a time complexity of O(log(n)).

B-Tree: The advantage of a B-tree over a binary search tree comes as a result of hardware. In the case that searches are implemented on a hard disk rather than memory it is advantageous to use a B-tree. Because read/write times impact a trees complexity a very deep tree is not beneficial in the case of hard disk. Because the B-tree is shallower and has a higher branching factor it is better to use a B-tree.

(5) Remember we discussed the implementation of queue and stack as circular arrays. What would happen for queue if we do not implement it as circular? And what would happen for stack? (10pts)

Queue is best implemented as a circular array as it helps avoid a need to shift every element upon modification. In the case that the head element of the structure is removed (dequed), to avoid a head of NULL it would be necessary to shift every subsequent element one place forward, rather than doing this we implement as circular and shift our pointer one place forward. By not implementing as circular we get the problem of shifting all elements forward on every deque.

Stack implemented as a circular array is non feasible, the advantage that circular arrays offer is their ability to simplify the implementation of FIFO structures. Stack is a LIFO data structure, because of this the only time the first element in the array is being popped is when it is the only element in the array, and even in this case the problem of a NULL head doesn't present. Stacks implementation should not be done using circular, the result of stack being circular would be no LIFO, as such a stack would be correctly implemented should it not be implemented as circular.