The data items listed below are to be used to answer the following parts to this question: [30 pts]
23 46 79 21 55 33 44 57 56

a) Create a binary search tree for the data items drawing a diagram to show the final tree after all items have been inserted. (5 points)
b) Write the contents of the tree using:
   1. inorder traversal (5 points)
   2. preorder traversal (5 points)
   3. postorder traversal (5 points)
c) Create an AVL tree for the data items. Show each stage of the tree as it gets built along with the single or double rotations (when needed). (10 points)
Recall that the abstract data type Queue has operations enqueue(data) (add data to the tail of the queue), and dequeue() (remove data from the head of the queue and return it). With a queue that holds characters, the input sequence AB (ordered from left to right), plus an operation output(character) (print next character from either the input sequence or the queue on standard output) you could generate (processing the input in left-to-right order):

\[ 18 \text{ pts} \]

A B output(A) output(B)  
B A enqueue(A) output(B) output(dequeue())

Suppose your input were ABC. Which of the six permutations of ABC can you generate using enqueue(data), dequeue(), and output(data)? Show the operations that yield the permutations that are possible. Explain why any impossible permutations are impossible (it's not enough to show you've tried lots of combinations of operations). To get you started, you can generate:

ABC output(A) output(B) output(C) Show how to produce the other 5 permutations ACB, BAC, BCA, CAB, CBA given the input ABC.

**Question No. 3**  
Marks : 5

Consider the following sequence of push operations in a stack:  

\[ 5 \text{ pts} \]

stack.push("1");  
stack.push("2");  
stack.push("3");  
stack.push("4");  
stack.push("5");  
stack.push("6");

You can insert as many stack.pop()'s as you like in the above sequence of stack.push's to get a desired output. Which of the following cannot be an output?

1 123456  
2 325416  
3 342561  
4 342615  
5 342165

**Question No. 4**  
Marks : 5

Which traversal gives a decreasing order of elements in a heap where the max element is stored at the top?  

\[ 5 \text{ pts} \]

1 pre-order  
2 in-order  
3 post-order  
4 level-order  
5 none of the above

**Question No. 5**  
Marks : 10

Let heap stored in an array as A = [3, 8, 4, 9, 11, 5, 6, 10, 12, 13]. Assume that the index of the first array cell is 1 and that this is also the root of the heap (i.e., element 3). In other words, the root of the heap contains the minimum element. What is the result of inserting 7 into this heap?  

\[ 10 \text{ pts} \]
Question No. 6  Marks : 12

For each part of the problem, name and justify one data structure that addresses the problem. Your answer should not be more than 3 lines. You may choose from the following list of data structures:

Stack Queue Binary search tree AVL tree Heap Hash table

(a) The data structure is initially empty. We then insert the values 2, 10, 12, 8, 6 and 4, in that order. Now, the only element we can remove is 12. (3 points)

(b) The data structure is initially empty. We insert the values 2, 4, 6, 8, 10, 1 and 7, in that order. If we want to remove an element, our only choice is 2 (3 points)

(c) The data structure initially contains n elements. We then insert elements 7, 14, 27, 68, and 3. We may now find any element, in average case, in constant time (i.e., O(1)) time (3 points)

(d) The data structure initially contains n elements. We then insert elements 2, 7, 5, 13, 11, 3 and 1, in that order. We may insert any element, in worst case, in time proportional to log n (i.e., O(log n)) time (3 points)

Question No. 7  Marks : 5

Suppose you are given a pointer “thisnode” that points to some node in a singly linked list (NULL terminated), as well as a stack which can hold pointers to list nodes. If x points to a node in the list, the member “x->next” holds a pointer to the node following x in the list. In 20 words or less, describe what the following pseudo code does: [5 pts]

```c
temp = thisnode->next
while (temp is not NULL)
{
    push temp onto the stack
    temp = temp->next
}
```
temp = thisnode
while(stack is not empty)
{
    temp->next = pop off stack
    temp = temp->next
}
temp->next = NULL