P.1) Answer the following questions:

(a) Each element in an array occupies 4 storage units. If the storage address of the 20^{th} element is 500, what is the starting address of the array? (Give the calculation process or explanation.)

Arrays have constant indexing as the different elements in the array have addresses varying by a constant value, which is 4 in this case. If the 20th element has an address of 500, then

a_n	a_{n+4}	a_{n+8}	$\dots$	a_{496}	a_{500}

is the array where n is the first memory address and a_{500} is the memory address of the 20th element. The address of an element in an array can be found by

$$\text{(Element Address)} = \text{(Starting Address)} + \text{(Element Index)} \cdot \text{(Storage per Element)}.$$

It is important to note that an Element's Index is its number in the array minus 1 as java is a 0-based index language. We can find the starting address by $500 = 19(4) + n$, and can solve to get $n = 424$. Therefore the starting address of the array is $\boxed{424}$.

(b) What do stacks and queues have in common?

Stacks and Queues are both linear data structures like arrays and linked lists and both store collections of elements. They are both dynamic in size, so they can dynamically grow or shrink in size as elements are added and removed over time. They can both be implemented using other data structures, and specify the order in which elements are processed. For example, stacks follow "Last In, First Out" (LIFO), while queues follow "First In, First Out" (FIFO). These behaviors cause them to be useful for many algorithms.

(c) For a binary tree with a depth of 8, what are the maximum and minimum possible numbers of nodes it can have? (Give the calculation process or explanation.)

A minimal binary tree has only one node per level as each node only has one child. Therefore assuming the parent node is at depth 0, then there can be a minimum 9 nodes in a binary tree with depth $\boxed{8}$.

A full binary tree, on the other hand, has 2 children per node. The number of nodes on any given level of a binary tree is given by 2^h, where h is the current level on the tree (0-based index). Therefore the total number of nodes is given by $2^0 + 2^1 + 2^2 + \cdots + 2^8 = \sum_{n=0}^{8} 2^n = \boxed{511}$.

(d) A complete binary tree contains 256 nodes. What is its depth, and how many leaf nodes does it have? (Give the calculation process or explanation.)

Generally, the total number of nodes in a tree can be found without using a summation as $N_{max} = 2^{d+1} - 1$, where d is the depth of the tree. This is derived from the geometric series relation $S = \frac{a_1 r^n - 1}{r-1}$ where n is the total number of terms (the depth plus 1).

We can solve for the depth of the tree by taking the floor function of d after solving for it as $d = \lceil \frac{\ln(N+1)}{\ln 2} - 1 \rceil$. Given that $N = 256$, we find that $\boxed{d = 8}$. We can solve for the total number of leaf nodes by observing that in a full binary tree with a depth of 7, there are $2^7 = 128$ leaf nodes. Also, a tree with a depth of 7 has 255 total nodes, and ours has 256. Therefore one of the nodes on the 7th level has a child, but there are still a total of $\boxed{128}$ leaf nodes on the tree.

(e) If the input sequence of a stack is 1, 2, 3, 4, 5, 6, provide three possible output sequences.

1. Push all → pop all.

$6, 5, 4{,}3, 2, 1$	Push all elements
Output Sequence: $6, 5, 4{,}3, 2, 1$	Pop all elements

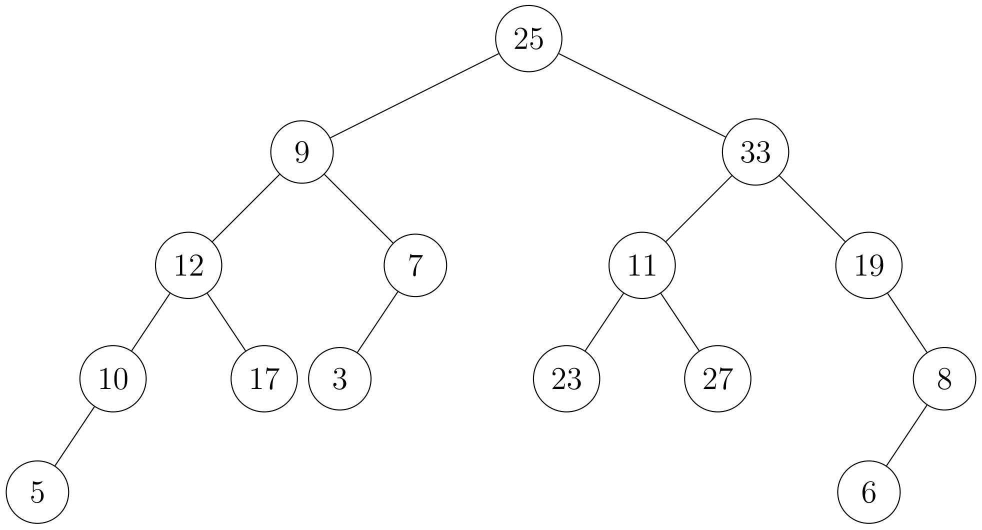
2. Pop midway through pushing elements.

$3, 2, 1$	Push 1, 2, 3 onto the stack
2,1	Pop 3
$6, 5, 4, 2, 1$	Push the rest of the elements
Output Sequence: $3, 6, 5, 4, 2, 1$	Pop the rest of the stack

3. Alternate pushing and popping.

2,1	Push 1, 2 onto stack
1	Pop 2
$4, 3, 1$	Push 3, 4 onto stack
3,1	Pop 4
$6, 5{,}3, 1$	Push 5, 6 onto stack
Output Sequence: $2, 4, 6, 5, 3, 1$	Pop the rest of the stack

P.2) Consider the following binary tree with 15 nodes. Describe the order of nodes visited in each of the following traversals of the tree:



(a) In-Order
Nodes are visited in Left-Root-Right order. This gives:

$$\boxed{5, 10, 12, 17, 9, 3, 7, 25, 23, 11, 27, 33, 19, 6, 8}$$

(b) Pre-Order
Nodes are visited in Root-Left-Right order. This gives:

$$\boxed{25, 9, 12, 10, 5, 17, 7, 3, 33, 11, 23, 27, 19, 8, 6}$$

(c) Post-Order
Nodes are visited in Left-Right-Root order. This gives:

$$\boxed{5, 10, 17, 12, 3, 7, 9, 23, 27, 11, 6, 8, 19, 33, 25}$$

P.3) Convert the following expressions from infix into postfix or from postfix into infix. *(Use PEMDAS to convert expressions)*

(a) 7 / 5 * G ˆ 6 * C + A ˆ 3 * B + Z - T
This expression is in infix, and should be converted to postfix as follows:

$$\boxed{\text{7 5 / G 6 ˆ * C * A 3 ˆ B * + Z + T -}}$$

(b) $X * Y^2 * \frac{7}{\sqrt{1-\frac{A^2}{B^2}}} - Z$
We can first rewrite this so that its easier to work with as:

X * Y ˆ 2 * 7 / (1 - A ˆ 2 / B ˆ 2) ˆ 0.5 - Z

This expression is in infix, and should be converted to postfix as follows:

$$\boxed{\text{X Y 2 ˆ * 7 * 1 A 2 ˆ B 2 ˆ / - 0.5 ˆ / Z -}}$$

(c) X Y Z - / A 8 O P Q / - * G 2 ˆ * / -
This expression is in postfix, and should be converted to infix as follows:

$$\boxed{\text{X / (Y - Z) - A / (8 * (O - P / Q) * G ˆ 2)}}$$

(d) b 2 - 4 a c * - 2 a * / + 7 c * +
This expression is in postfix, and should be converted to infix as follows:

$$\boxed{\text{b - 2 + (4 - a * c) / (2 * a) + 7 * c}}$$