Part A (Klevis Tefa)

Question A.1

Target	Linear Search	Improved Linear Search	Binary Search
Algorithm	1	1	4
Computer	3	3	4
heap	8	8	1
int	9	9	4
open	15	15	4
bit	15	2	4
stack	15	15	4
queue	15	15	4
n = 100	100	100	7
n = 1000	1000	1000	10
n = 10^6	10^6	10^6	20
n = 10^9	10^9	10^9	30

Question A.2

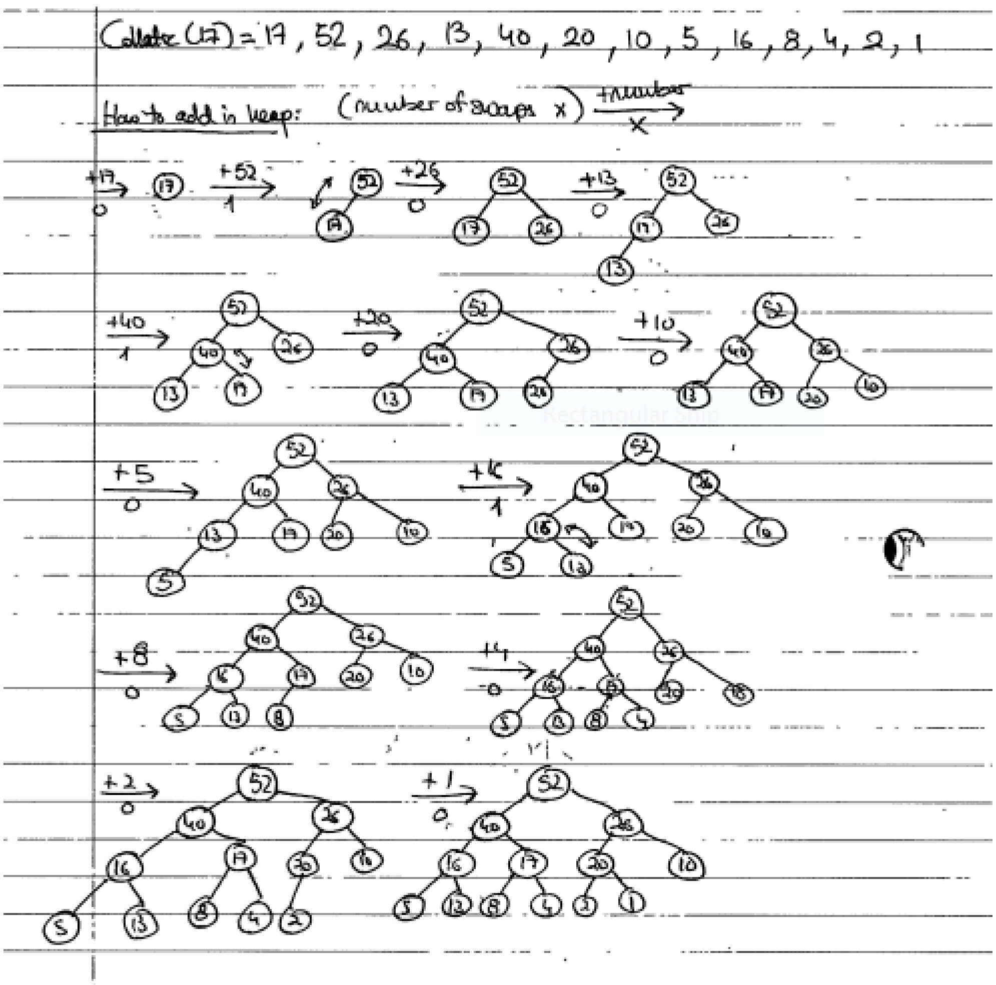
(a) 6 different almost level trees.

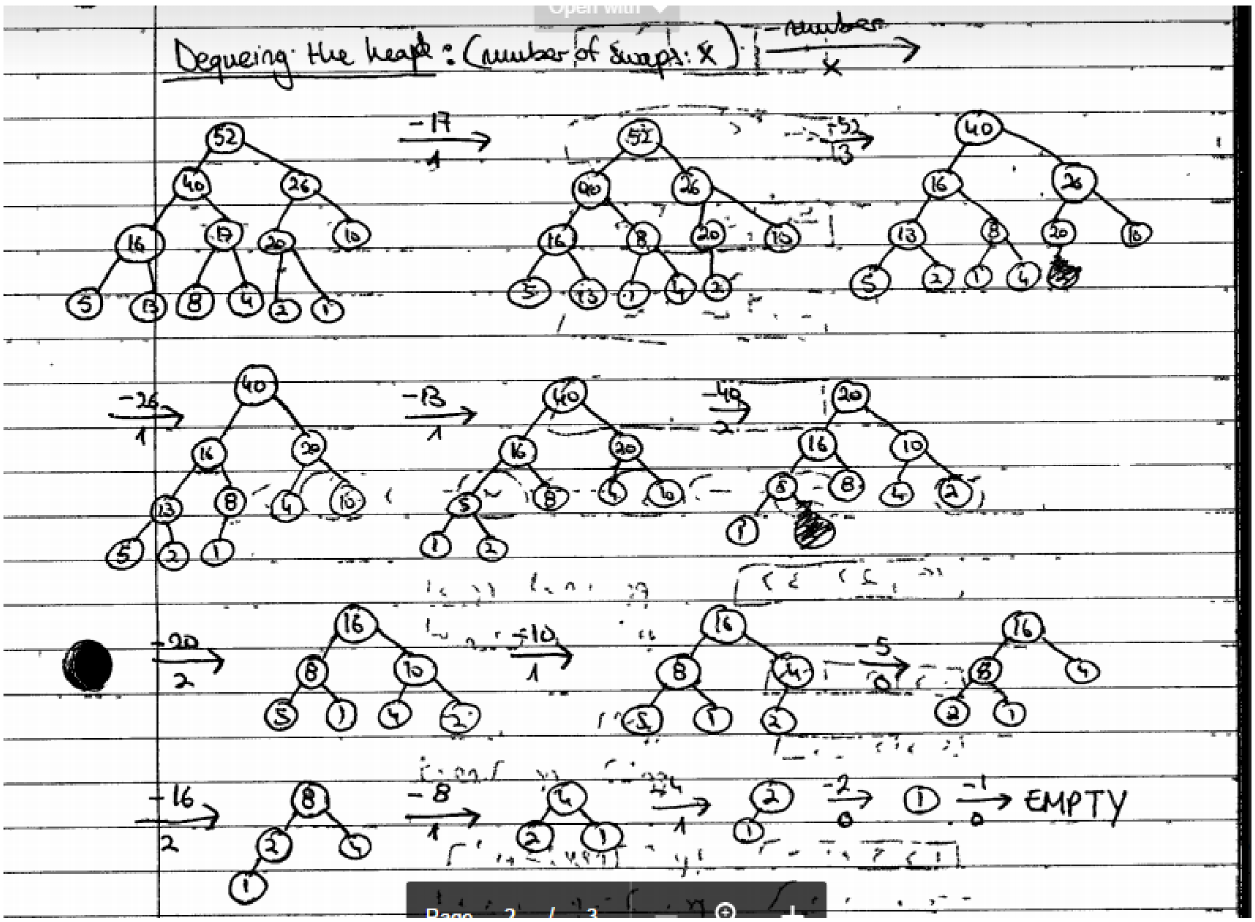
(b) Let L be the number of leaves, N the number of internal nodes, and T the total number of nodes in a full binary tree. Since each node is either an internal node or a leaf in full binary tree than it's trivial that T = N + L. From observing the structure of different full binary trees I came to the relation that L = N + 1. We can prove this by induction.
Proof: Let S = N (set of all integers ≥ 0) such that if a tree is a full binary tree with I internal nodes than L = N + 1.
(Base case): If N = 0, then the tree has only the root with no children since the tree is full. Hence there is only one leaf (i.e L = N + 1 = 0 + 1 = 1).
Now suppose for some integer K ≥ 0, every N from 0 through K is in S. (i.e: if a nonempty binary tree has N internal nodes (0 ≤ N ≤ K), then that tree has N + 1 leaves.
Let's have a tree with K + 1 internal nodes. Then the root of that tree will have two subtrees L and R, and suppose L has N_L internal nodes and R has N_R internal nodes (neither L nor R can be empty). So every internal node in L and R is an internal node in our original tree plus the root of the tree itself. Hence K + 1 = N_L + N_R + 1.
Now by induction hypothesis, L must have N_L + 1 leaves, and R must have N_R + 1 leaves. Since every leaf in our original tree is either in L or in R we have a total of N_L + N_R + leaves.
Therefore we must have K + 2 leaf nodes so K + 1 is in S. Hence by mathematical induction S = [0, ∞).
Since we proved that L = N + 1 (or N = L – 1) and we know that T = L + N (or T = L + L – 1 = 2L – 1).
Therefore 99 = 2L – 1 which means that L = 50. So we have 50 leaf nodes.

(c) Formula for a geometric series of the form:
$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{i=0}^{n-1} ar^i = a\frac{1-r^n}{1-r}$
In our case $r = 2$, $a = 2^0 = 1$, and $n - 1 = 29$ or $n = 30$. Therefore our required sum is
$1 * \frac{1-2^{30}}{1-2}$ = $2^{30} - 1$

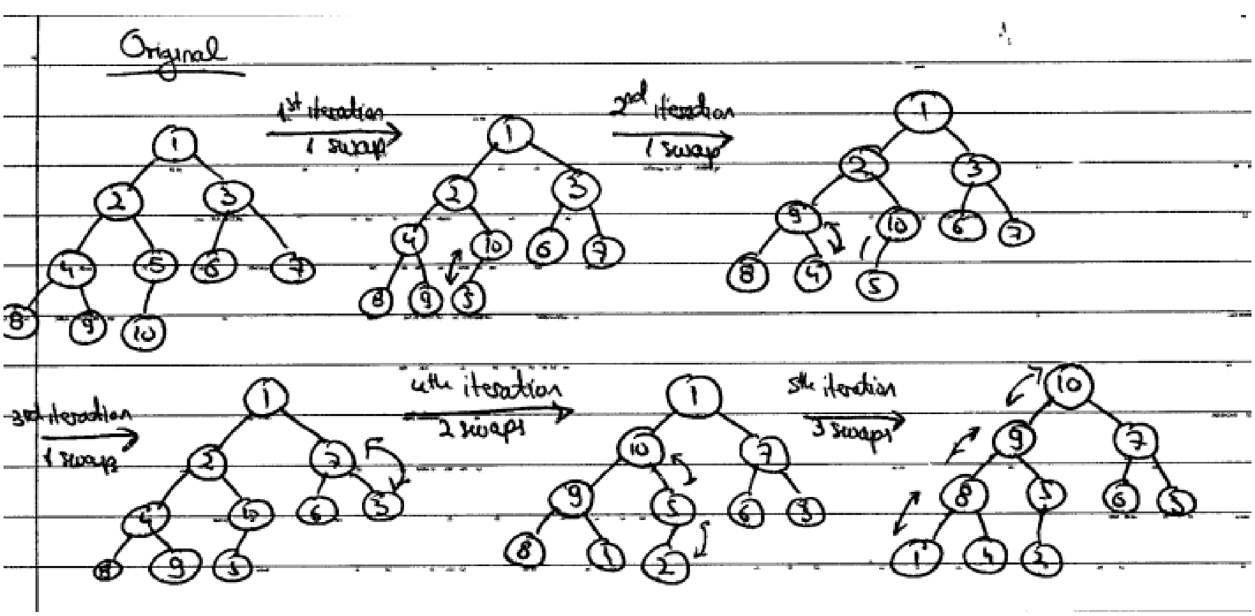
(d) According to Lecture 11, at level i, we have 2^i nodes. In the last level i = height – 1. In order to find height of a complete binary tree of 1000 nodes we can use the formula from lecture 11, where $height(n) = [\log(1+n)] - 1$. Hence, $height(1000) = [\log(1+1000)] - 1 = [\log(1001)] - 1 = 10 - 1 = 9$. So at the last level i = 8, therefore there are 2^8 = 256 nodes.

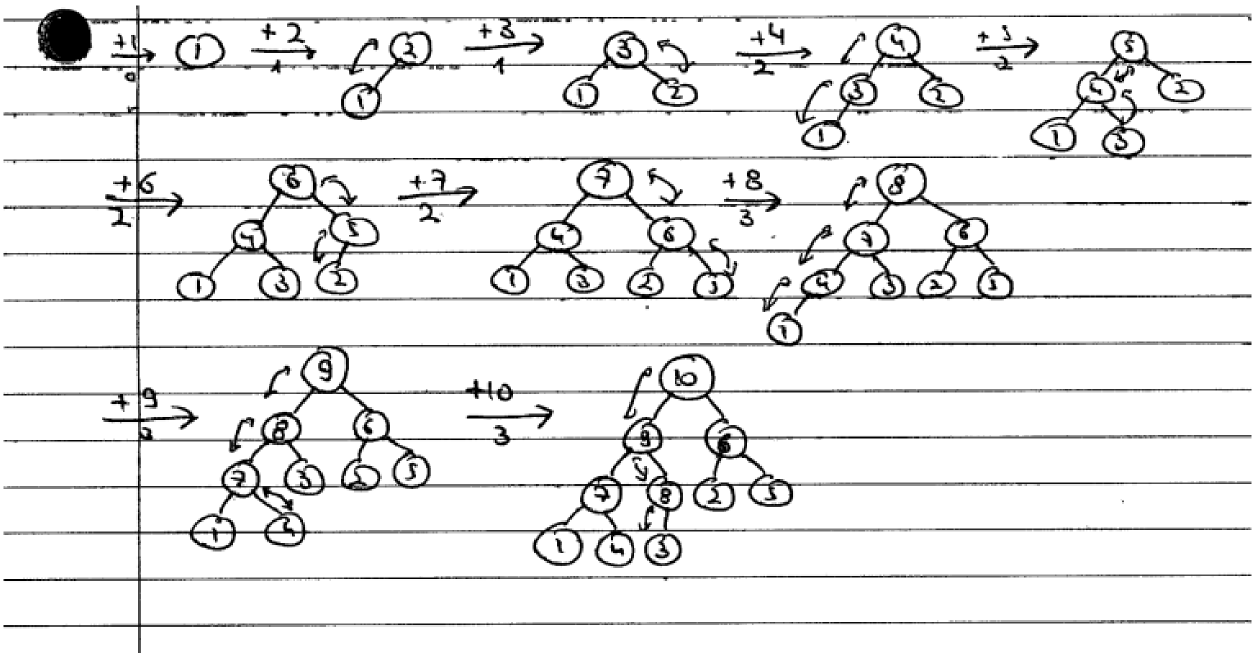
Question A.3




Dequeing the heap : (number of swaps: x)
EMPTY

Question A.4


Original
1st iteration
1 swap
2nd iteration
1 swap
3rd iteration
1 swap
4th iteration
2 swaps
5th iteration
3 swaps



c) Total number of swaps in part a) was 8 and in part b) it was 19. For this particular heap of size 10 the code in part a) is better since we have to do less swaps.