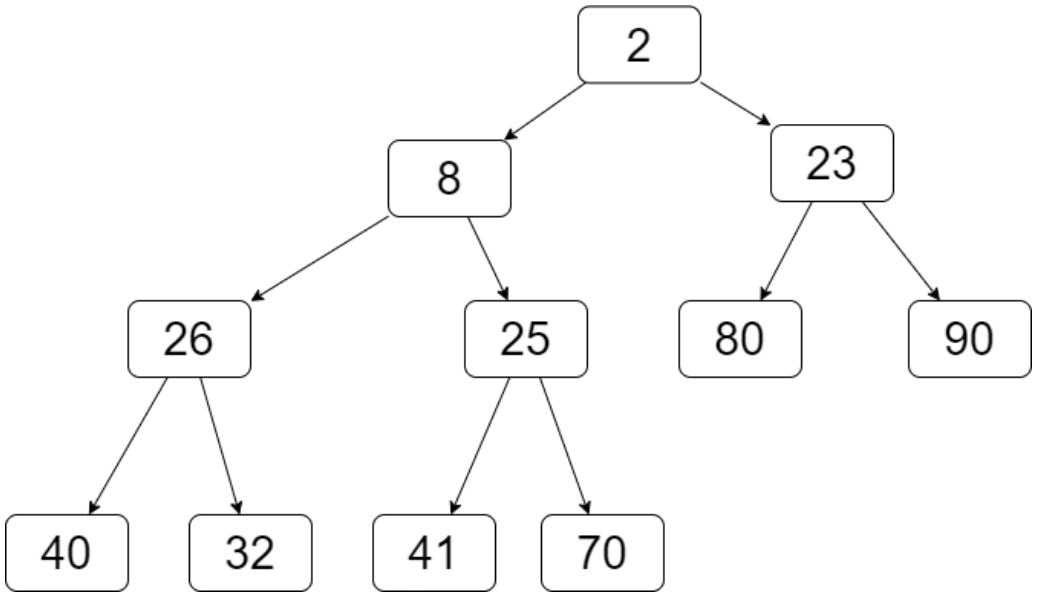
Question 1

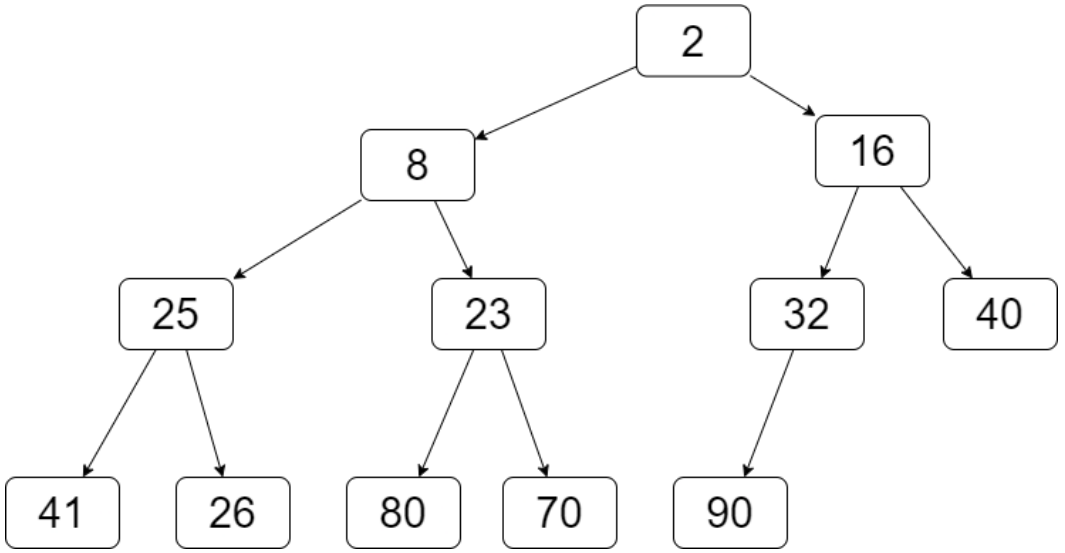
a)



b)

8	23	16	25	70	32	40	41	26	80	90

c)

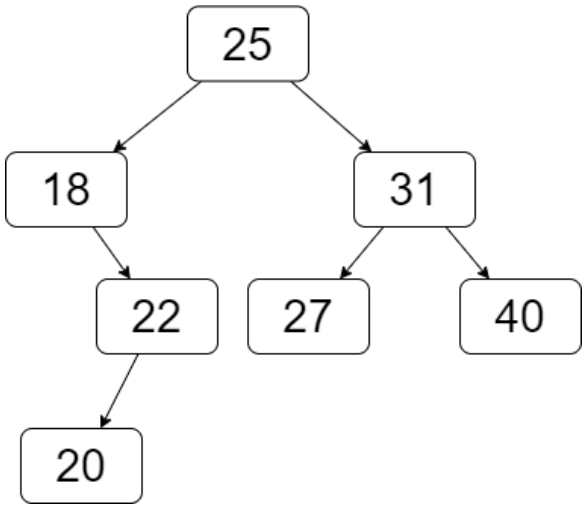


Swap with the parent until 16 becomes smaller than the parent

d) The insertion operation has a worst-case complexity of O(log n)

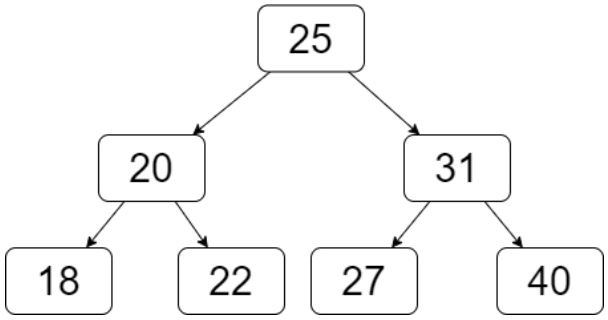
Question 2

a)

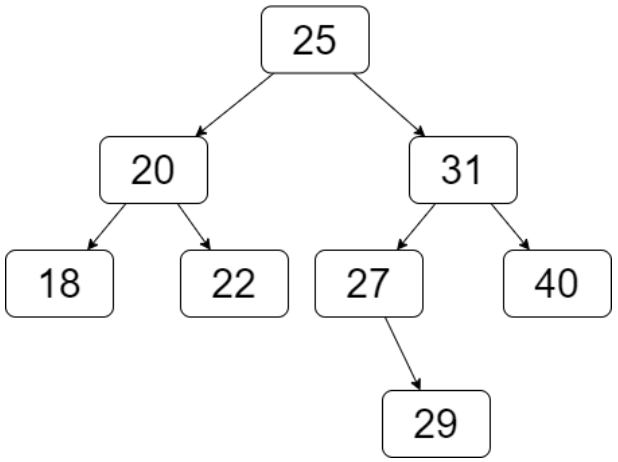


b) The imbalance is at Node 22

c)



d)



Question 3

a) Each colour represents a split group of numbers.
The underlined numbers represent an action (i.e. splitting numbers into groups or ordering by smallest number)

0	52	34	19	80	7	29	44	32	19	28
1	**52**	**34**	**19**	**80**	**7**	**29**	**44**	**32**	**19**	**28**
2	**52**	**34**	**19**	80	7	29	44	32	19	28
3	**52**	**34**	19	80	7	29	44	32	19	28
4	**52**	34	19	80	7	29	44	32	19	28
5	**34**	**52**	19	80	7	29	44	32	19	28
6	**19**	**34**	**52**	80	7	29	44	32	19	28
7	**19**	**34**	**52**	80	7	29	44	32	19	28
8	**19**	**34**	**52**	7	80	29	44	32	19	28
9	**7**	**19**	**34**	**52**	**80**	**29**	**44**	**32**	19	28
10	7	19	34	52	80	**29**	**44**	32	19	28
11	7	19	34	52	80	**29**	44	32	19	28
12	7	19	34	52	80	**29**	**44**	32	19	28
13	7	19	34	52	80	**29**	**32**	**44**	19	28
14	7	19	34	52	80	**29**	**32**	**44**	19	28
15	7	19	34	52	80	**29**	**32**	**44**	19	28
16	7	19	34	52	80	**19**	**28**	**29**	**32**	**44**
17	**7**	**19**	**19**	**28**	**29**	**32**	**34**	**44**	**52**	**80**

b) i. Linear Search will take 9 comparisons to find 52
ii. Binary Search will take 3 comparisons to find 52

Question 4

a) U → V → Q → L → G → L → M → H → I → D → E
Data Structure: Stack

b) U, V, Q, L, R, G, M, K, S, W, H, N, F, P, X, I, A, Y, D, B, E
Data Structure: Queue

Question 5

a) To resolve collisions, Chaining combines a linked list and a hash table. When two or more elements are hashed to the same location, they are represented in a chain-like single-linked list.

b) With the Linear Probing technique, a sequential search is processed to find an empty location when collisions occur. The idea is to place a value in the next available position.

c)

11 = 0 * 20 + 11
1 probe

28 = 1 * 20 + 8
1 probe

21 = 1 * 20 + 1
1 probe

2 = 0 * 20 + 2
1 probe

75 = 3 * 20 + 15
1 probe

88 = 4 * 20 + 8
2 probes

19 = 0 * 20 + 19
1 probe

54 = 2 * 20 + 14
1 probe

34 = 1 * 20 + 14
3 probes

d) Advantages

- Requires much less memory
- Less complex and easier to implement

Disadvantages

- Primary clustering occurs when there are huge blocks of occupied cells in the hash table.
- In linear probing, values tend to cluster, making the probe sequence longer.

Question 6

a)

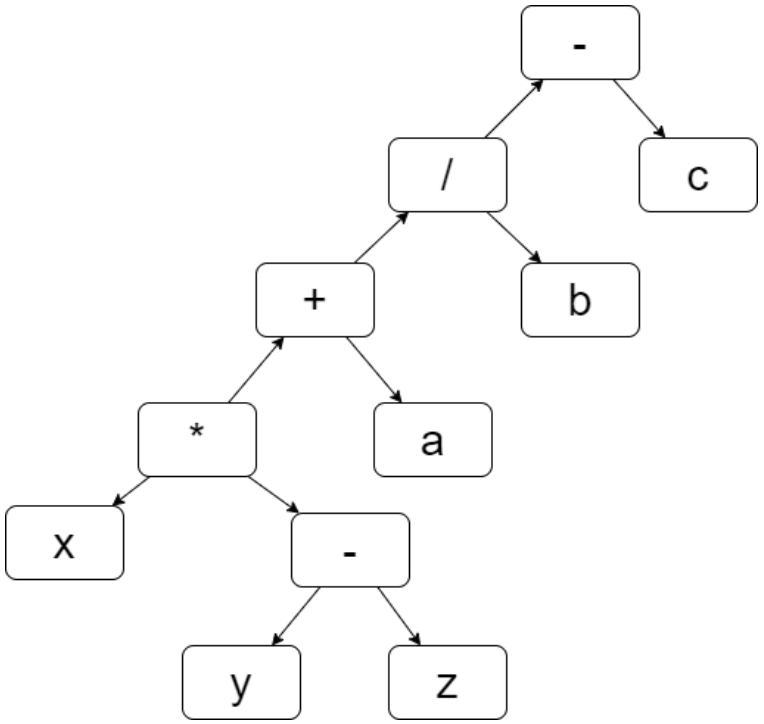
	a	b	c	d	e
a	0	1	1	1	0
b	1	0	0	0	1
c	1	0	0	1	1
d	1	0	1	0	1
e	0	1	1	1	0

b) Dijkstra's Algorithm works by dividing the vertices into two sets. One is the selected set, one is the candidate set. At each step, we move the cheapest node to reach from the candidate set to the selected set.

$A \rightarrow D \rightarrow C \rightarrow E \rightarrow B$

Question 7

a)



b) x y z - * a + b / c -

c)

	Expression	Stack
0	x	x
1	y	x y
2	+	(x+y)
3	y	(x+y) y
4	z	(x+y) y z
5	w	(x+y) y z w
6	*	(x+y) y (z*w)
7	-	(x+y) (y-(z*w))
8	*	((x+y)*(y-(z*w)))

Question 8

a) Dynamic Programming involves breaking a problem into similar subproblems. To solve a problem in Dynamic Programming, the solutions of overlapping subproblems are saved in a data structure. This process is known as Memoization

If a problem has Optimal subproblems - where a larger problem can be broken into similar subproblems - and overlapping subproblems - where the recursive solution to the problem contains subproblems that are repeated - it can be solved.

Fibonacci Sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 24

We create a function fib(n) that returns the n^{th} number of the Fibonacci series. To solve the problem recursively, we put the following formula inside a function:

fib(n) = fib(n-1) + fib(n-2) if n > 1

Using the function, we can store the results of previously solved subproblems in a list-like data structure. The function will determine whether or not a subproblem has already been addressed. If it has already been solved, it does not need to be solved again. When a subproblem is solved, it is added to a list (Memoization).

First, the list must be filled with a value that is never a solution (i.e. -1). The list will be populated with -1 for a specified maximum length. For fib(n), we verify whether list[n] equals -1 or not. If it is -1, the problem has never been solved, so we will solve it. If it is not -1, the problem has been solved, and we will simply return it.

b) Algorithm change(n)

change(n) = min(change(n-2), change(n-3), change(n-5)) + 1