PUT - KOE-045

Data Structure

KAVISH AGRAWAL
1802731076

Section - B.

Question 11.

A + B - C ^ D * E $ F + G / H - I + J

Symbol	Stack	Exp.
A		A
+	+	A
B	+	AB
-	-	AB+
C	-	AB+C
^	- ^	AB+C
D	- ^	AB+CD
*	- *	AB+CD^
E	- *	AB+CD^E
$	- * $	AB+CD^E
F	- * $	AB+CD^EF
+	+	AB+CD^EF$*-

Kavish

G	+	AB + CD^ EF $ * - G
/	+/	"
H	+/	AB + CD^ EF $ * - G H
-	-	AB + CD^ EF $ * - G H / +
I	-	AB + CD^ EF $ * - G H / + I
+	+	AB + CD^ EF $ * - G H / + I * -
J	+	AB + CD^ EF $ * - G H / + I - J

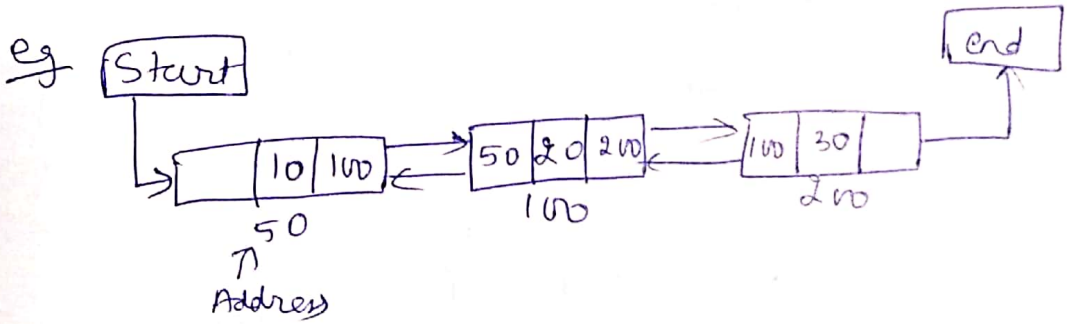
Postfix : ⇒ AB + CD^ EF $ * - G H / + I - J +

Ans

Question 12.

Doubly linked list

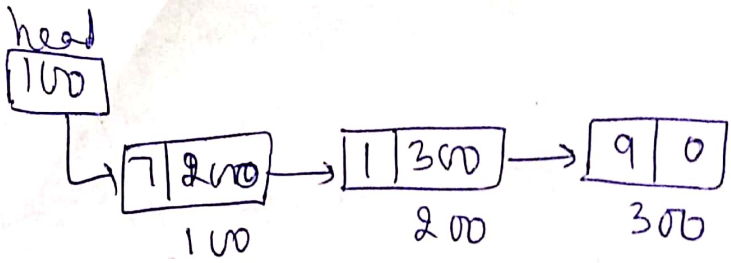
In a doubly linked list, each node contains a data part and two addresses, one for previous node and one for the next node.



Kanish

To insert an element at begin

For eg we have list,



Step 1:

```
struct node
{ int data;
  struct node *next;
};
struct node *head, *newnode;
```

Step 2: Allocate memory

```
newnode = (struct node *) malloc (sizeof (struct node));
```

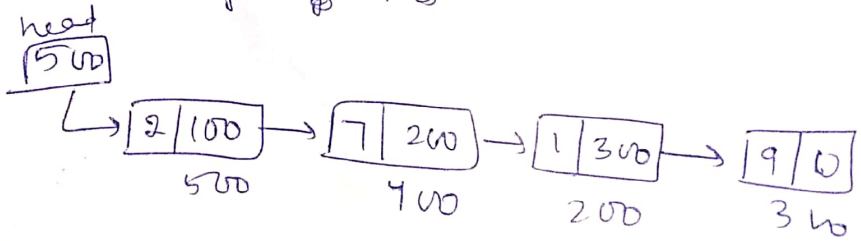
Step 3: Take new node as input

```
printf ("Enter number");
scanf ("%d", &newnd->data);
newnode->next = head;
head = newnode;
```

~~Step 4~~

// Here we are setting the address for new node in head and the address of previous 1st node in newnode.

Therefore, final list is -



Kavish

Question 13

Binary Search

→ In this method, the sorted array is devided into two parts and we compare the key with mid element of array.

→ This comparison results either in the match between key and the mid element or identifying the left half or the right half of the array to which the desired element may belong.

→ When the current element ≠ arr[mid], Repeatedly devide the array again.

In this way, either the element is detected at mid or the division leads to a half consisting of no element.

→ Time complexity of Binary search is $O(\log n)$.

Kavish

C- Procedure

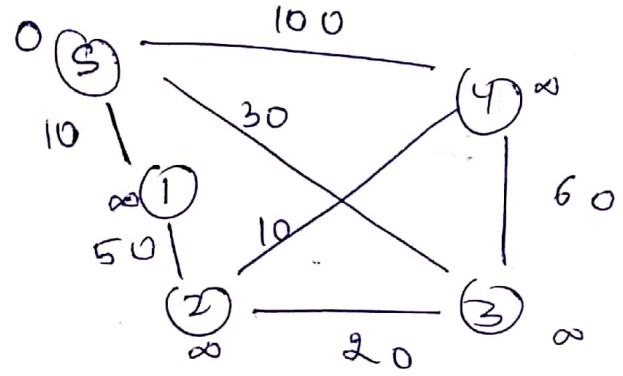
```
int BinarySearch(int a[], int n, int key)
{ int low, high, mid;
  low = 0;
  high = n-1;
  while( low <= high )
  { mid = (low + high)/2;
    if (key == a[mid])
      return mid;
  }
}
```

Kavish

Section - C

Question 16. b) Part

~~Set~~ set the distance of S as 0 and rest as ∞.



By using condition, if (d(u) + c(u,v) < d(v))

~~d(1) =~~ For (1) $0 + 10 < \infty$ ✓

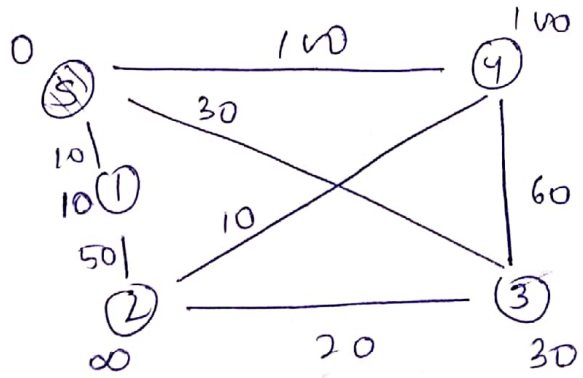
$\Rightarrow d(1) = 10$

For (3)

$d(3) = 30$

For (4) $\Rightarrow d(4) = 100$

2)

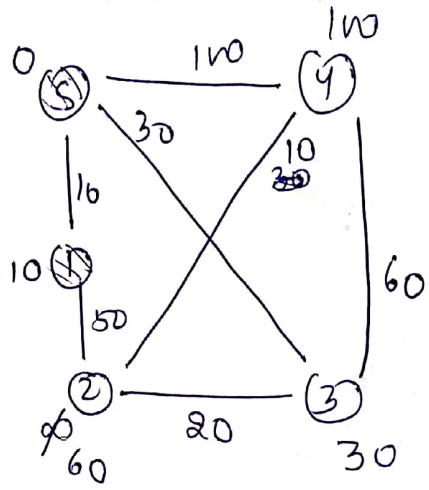


Kavish

Now from ①,

For ②, ∵ 10 + 50 < ∞ ✓

⇒ d(2) = 60



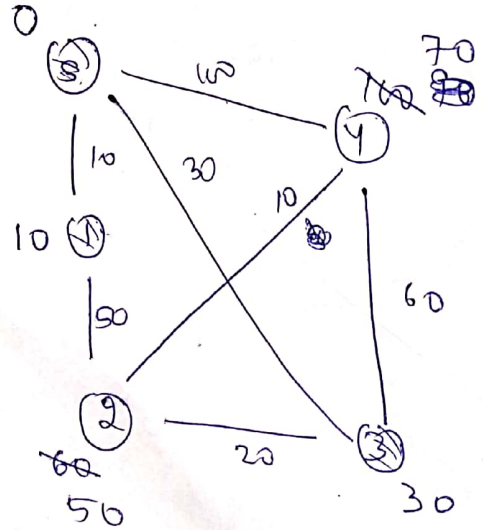
Now from ③

For ②, 30 + 20 < 60 ✓

d(2) = 50

For ④, 10 + 60 < 100

d(4) = 70



Kavish

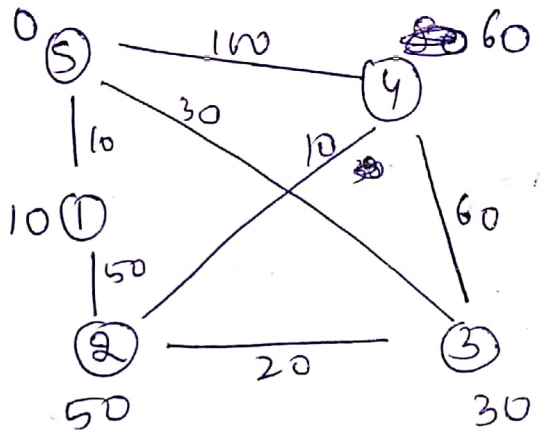
Now from ⑤,

For ④ ~~50 + 30 < 90~~ ✓

$d(4) = 80$

For ④, $50 + 10 < 70 \Rightarrow d(4) = 60$

So final Graph is :



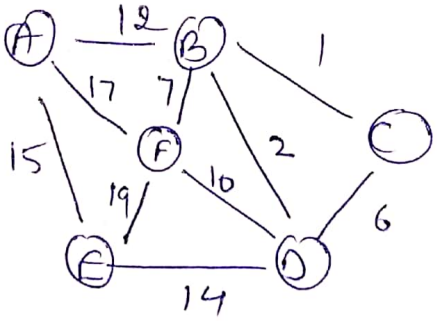
Any

Question 17 b) Part

Let the source be (A),

1) (A) —12— (B)

2)



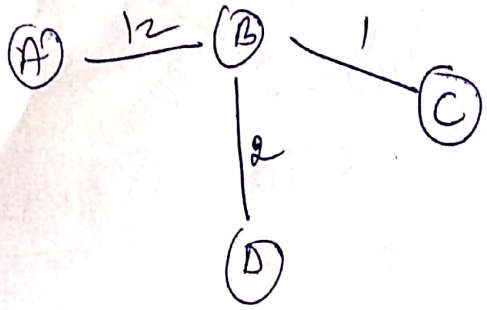
3) Now select minimum cost from either A or B, ie 1

4) (A) —12— (B) —1— (C)

Finish

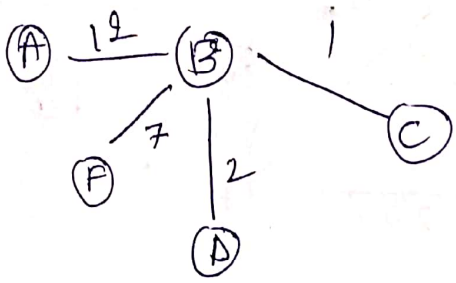
• Now ~~start~~ min cost from, A, B or C is (2)

>>



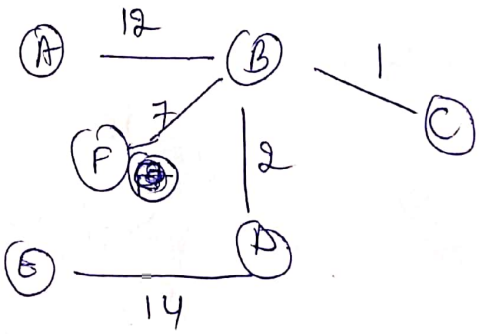
• Now Select (B) — (F)

>>



• Now at last To connect E, minimum cost is 14, ~~from~~

>>



This is the MST

Minimum cost = 12 + 1 + 2 + 7 + 14

= (29)

Kavish

Question 18.

a) Part

Insertion Sort

- It is like sorting a hand of playing cards start with an empty hand and the cards facing down the table.
- Pick one card at a time, and insert it into the correct position in the left hand.
- Compare it with each of the cards already in the hand, from right to left.
- The cards held in the left hand are sorted.

Algorithm —

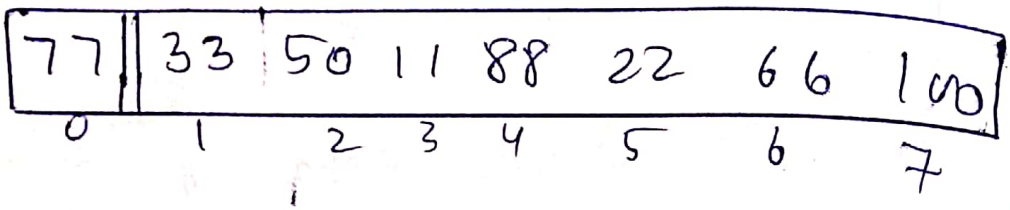
```
InsertionSort()
{  int i, key, j;
   for (i=1 ; i<n ;i++)
   { key = arr[i];
     j = i-1;
     while ( j>=0 && arr[j] > key )
     { arr[j+1] = arr[j];
       j--;
     }
     arr[j+1] = key;
   }
}
```

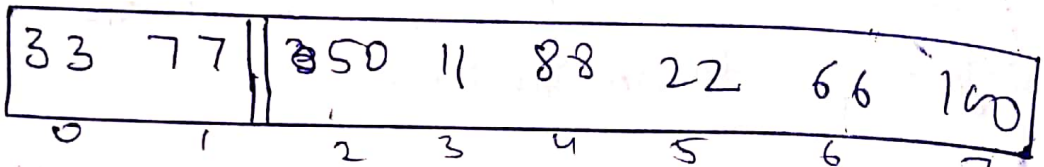
Kavish

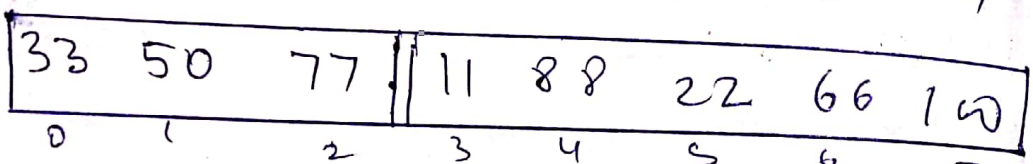
Given list;

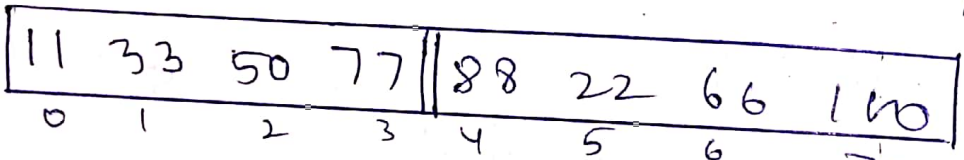
77, 33, 55, 11, 88, 22, 66, 100

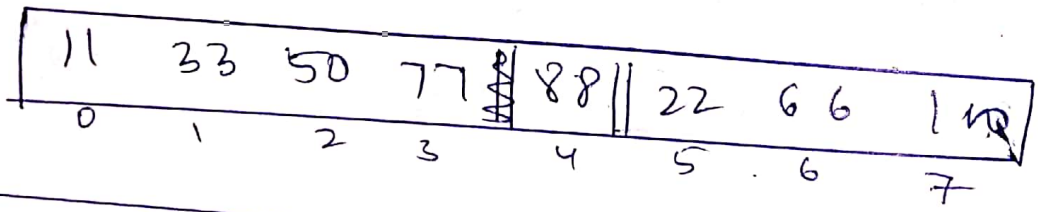
→)

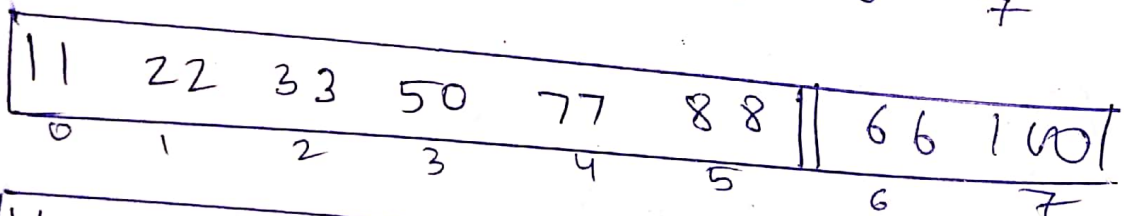


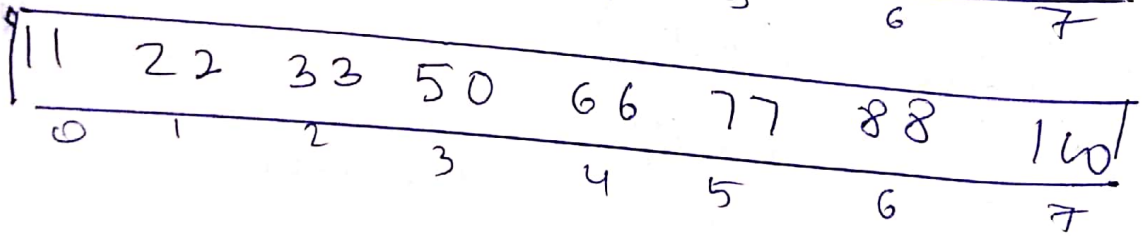












Kavish

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

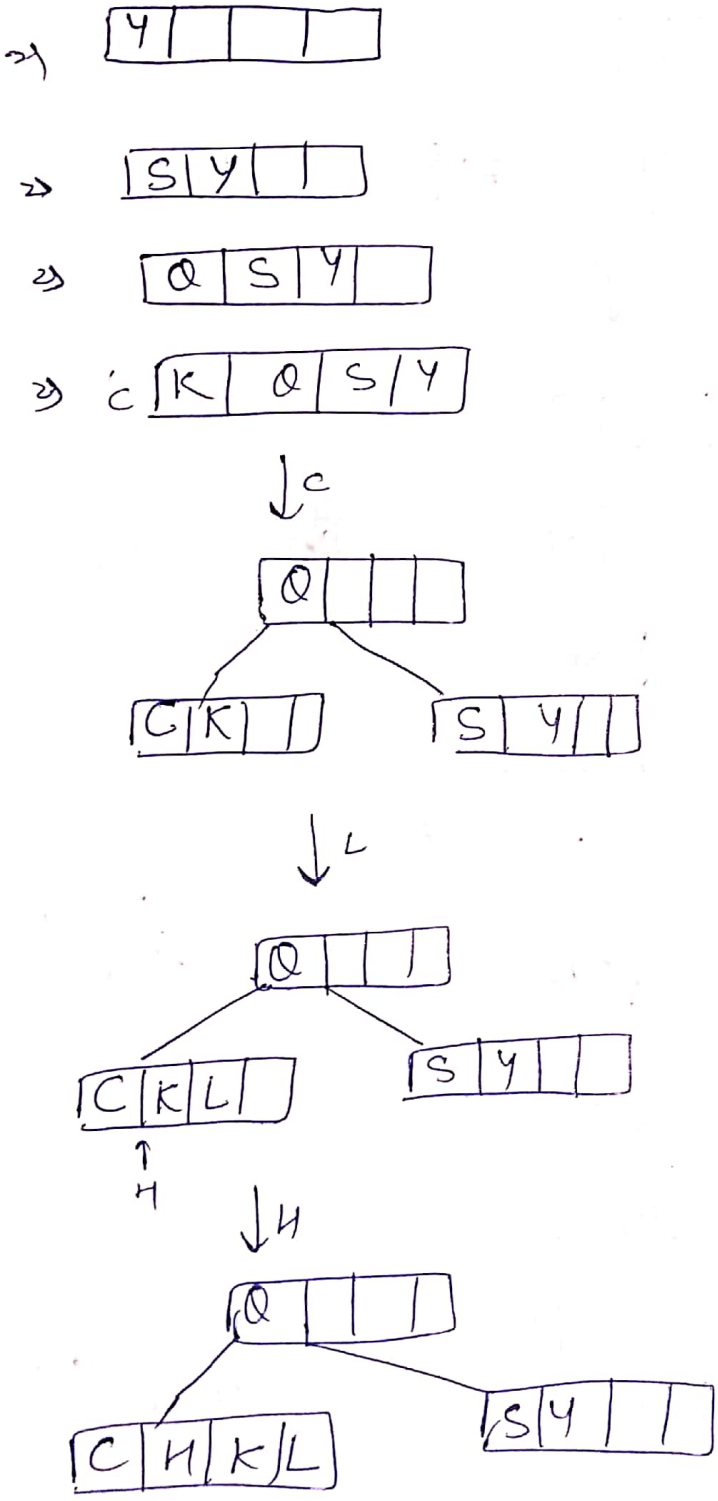
Question 19.

a) Part

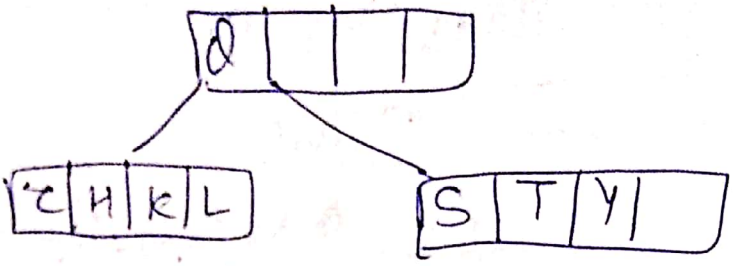
Y, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, Z

$m = 5$

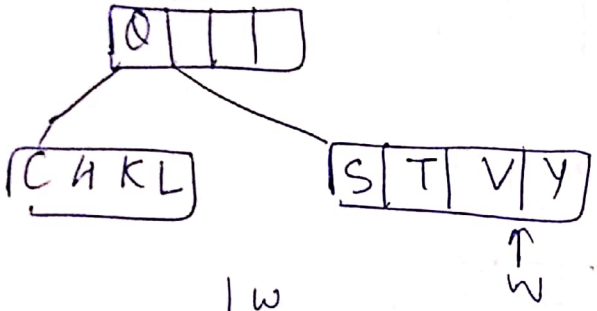
$Key = 5 - 1 = 4$



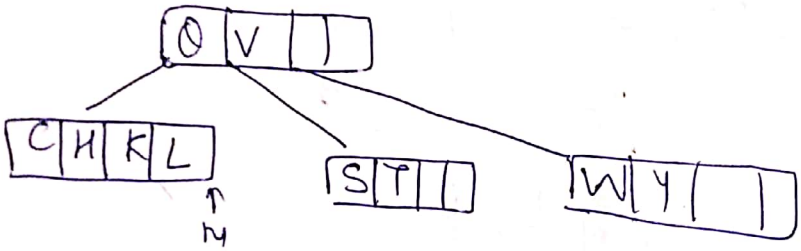
Kanish


Q
C H K L
S T Y

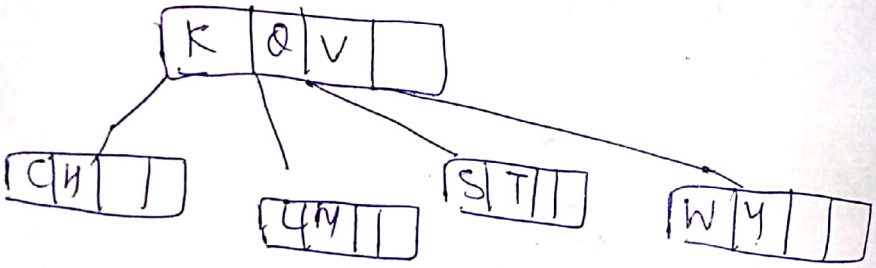
↓V


Q
C A K L
S T V Y
W

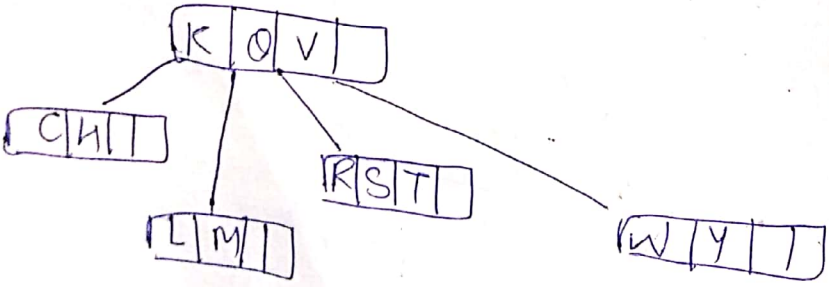
↓W


Q V
C H K L
S T
W Y
M

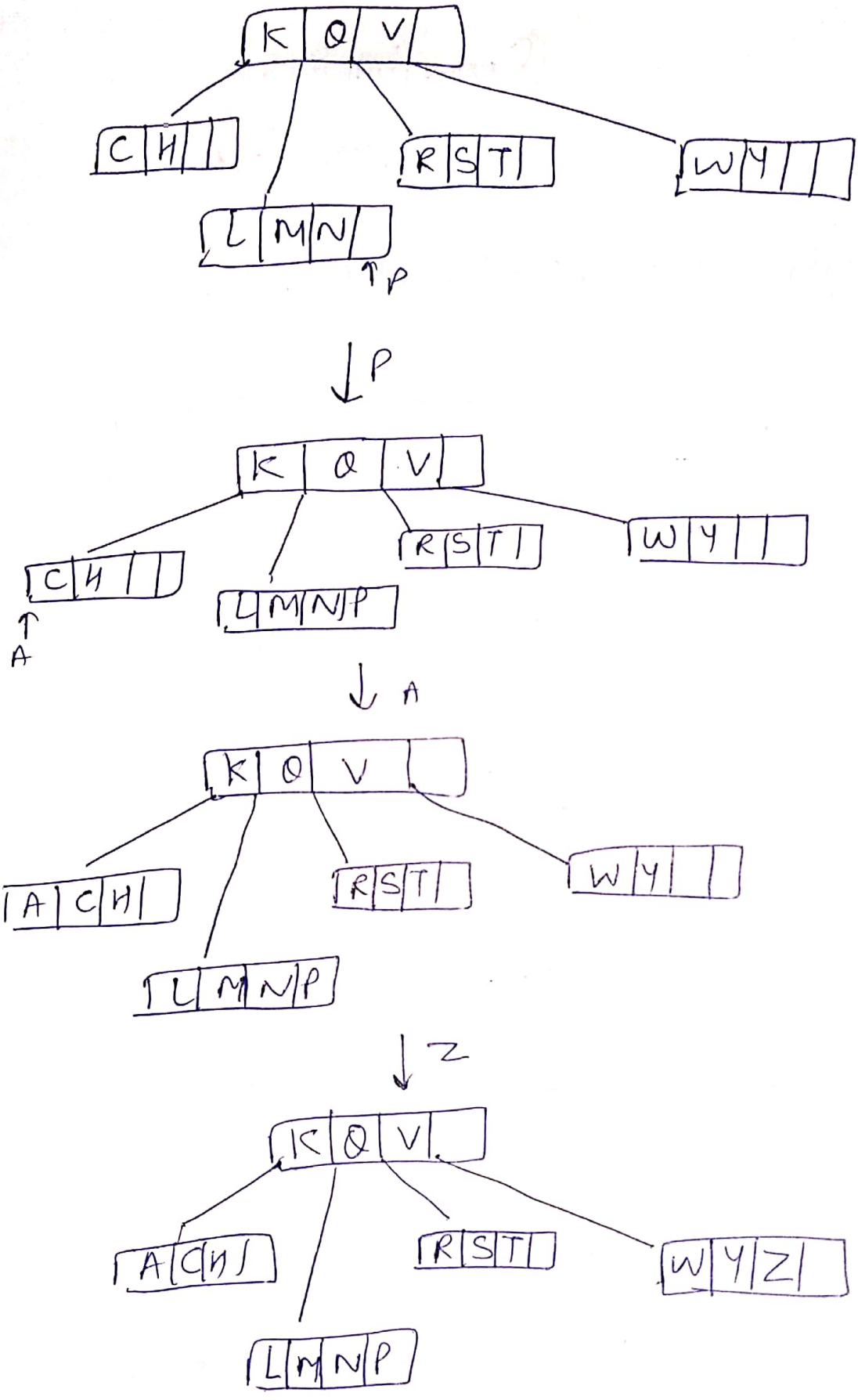
↓M


K Q V
C H
L M
S T
W Y

↓R


K Q V
C H
L M
R S T
W Y

↓N



B-Tree

Kavita

Question 20.

a) Part

Infix: $(a+b\wedge c\wedge d)*(e+f/d)$

Symbol	Stack	Reverse Polish Exp
(	(	
a	(	a
+	(+	a
b	(+	ab
^	(+^	ab
c	(+^	abc
^	(+^	abc^
d	(+^	abc^d
)	~~(+^~~	abc^d^+
*	*	abc^d^+
(	*(	〃
e	*(	abc^d^+e
+	*(+	abc^d^+e
f	*(+	abc^d^+ef
/	*(+/	〃

Kavish

d ⊕ *(+/ abc^d^+efd

) ⊕ abc^d^+efd/+

= ab^d^+efd/+*

Ans Reverse Polish Exp

Section - A

Question - 1

Infix to Post fix

Let X is an infix exp, Y be the equivalent postfix expression.

1. Scan X from left to right.
2. If an operand is encountered, add it to Y
3. If an operator is encountered,
   - Repeatedly pop from stack & add to Y each operator which has the same or higher precedance than operator.
   - Add operator to stack

Kavish