NAME KASHYAP SHAH
MOBILE 9408718809
A. YEAR 4CE2 B-BATCH
BRANCH COMPUTER ENGINEERING
18DCE115

Date
Page 1

08/06/2020

CE245 DATA STRUCTURE AND ALGORITHM

Q3 Advantages of circular queue over simple queue:

i) It takes up less memory than the linear queue
ii) A new item can be inserted in the location from where a previous item is deleted.
iii) Infinite number of elements can be added continuously but deletion must be used.

Enqueue in circular queue:

QINSERT (F, R, Q, N, Y)

1) if R=N
then R<1
else R<R+1

2) [OVERFLOW]
if F=R
then write ('overflow')
Return

3) [Insert element]
Q[R]<Y

4) [Is Front pointer properly set?]
if F=0
then F<1
Return

18DLE115

Date
Page 2

Dequeue in Circular Queue

QDELETE (F, R, Q, N)

i) IF F=0

then

write ('underflow')

Return (0)

ii) Y <- Q[F]

iii) if F=R

then F <- R <- 0

iv) if F=N

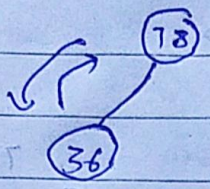
then F<-1

else F <- F+1

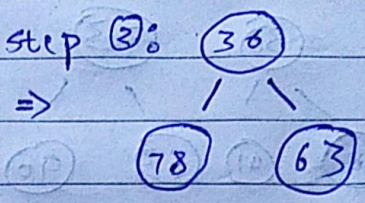
Return (Y)

Q8) 78, 36, 63, 41, 23, 22, 90, 74, 59, 44, 19

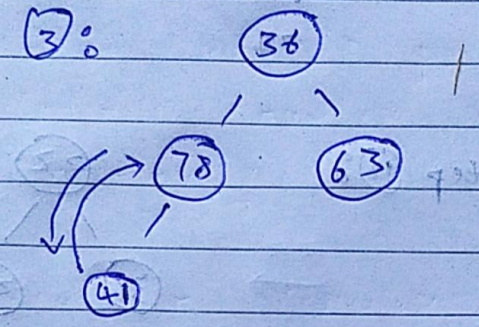
↳ step (1):



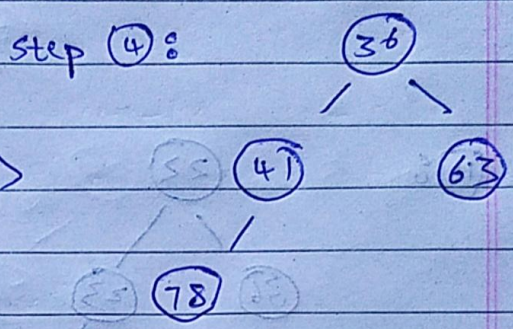
step (2):



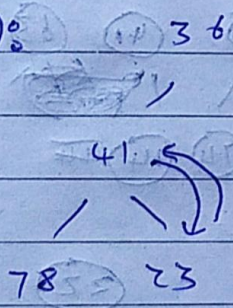
step (3):



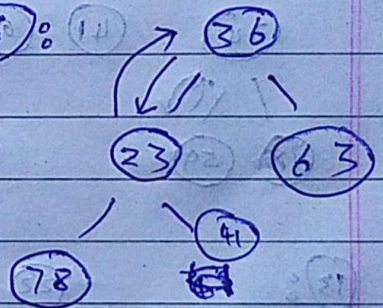
step (4):



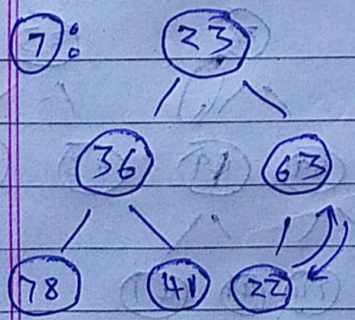
step (5):



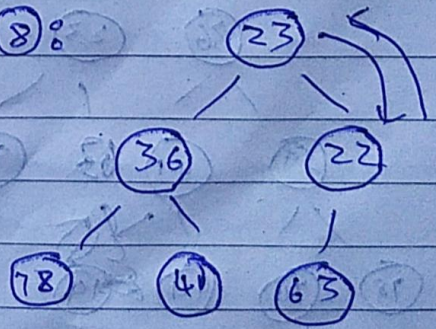
step (6):



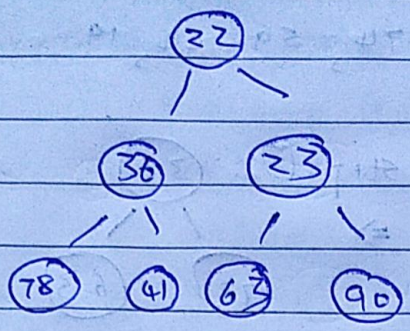
step (7):



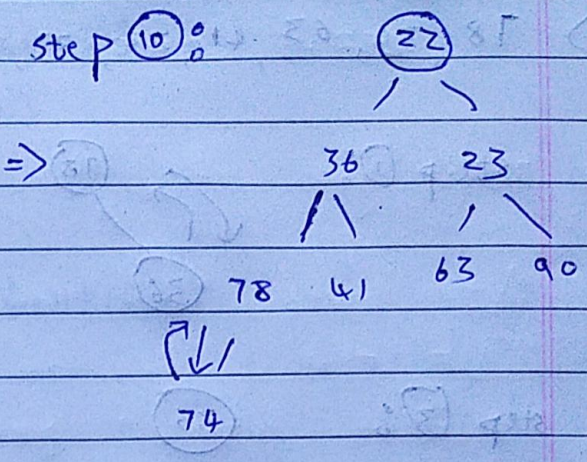
step (8):



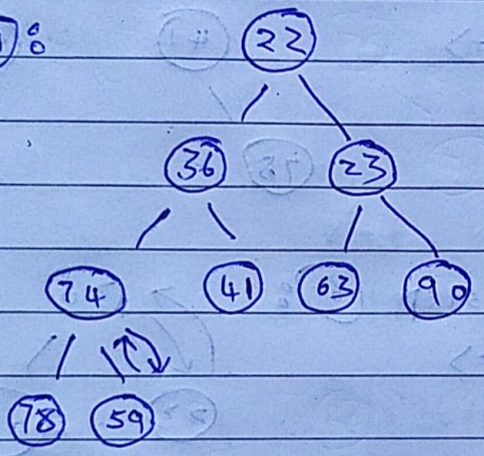
step (9):



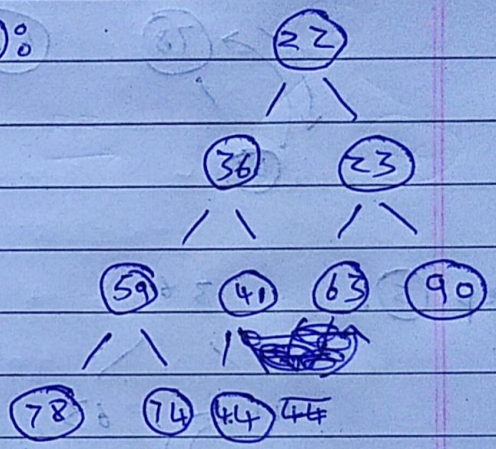
step (10):



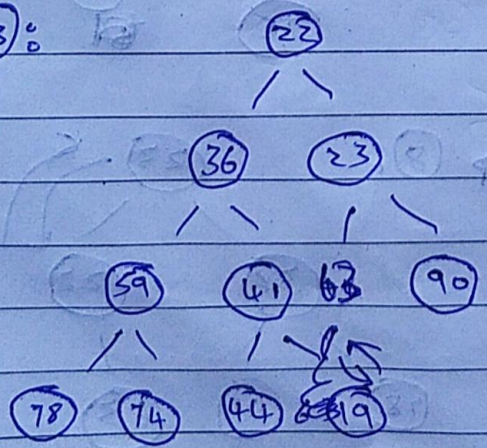
step (11):



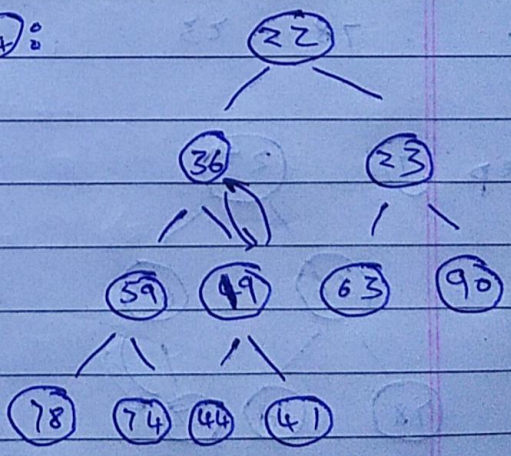
step (12):



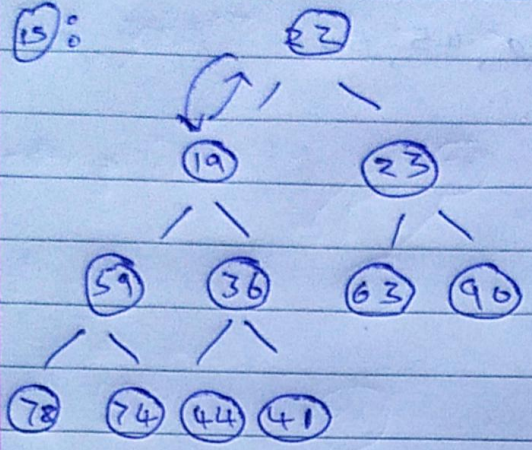
step (13):



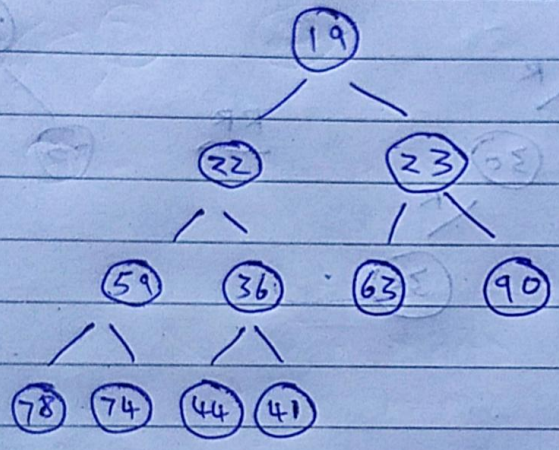
step (14):



Step 15 :



16



Q2) BFS sequence =>

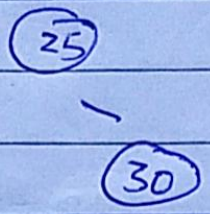
1 → 0 → 2 → 5 → 4 → 7 → 6 → 3

Yes, it is a connected graph

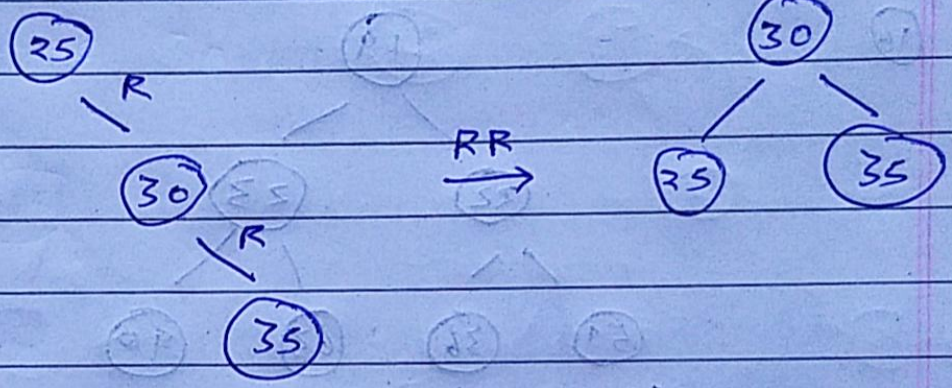
a1) Insert 25, 30, 35, 40, 45, 50

step 1 :

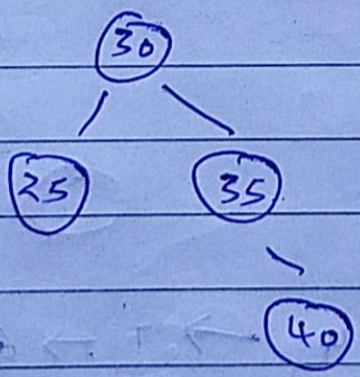
step 2 :



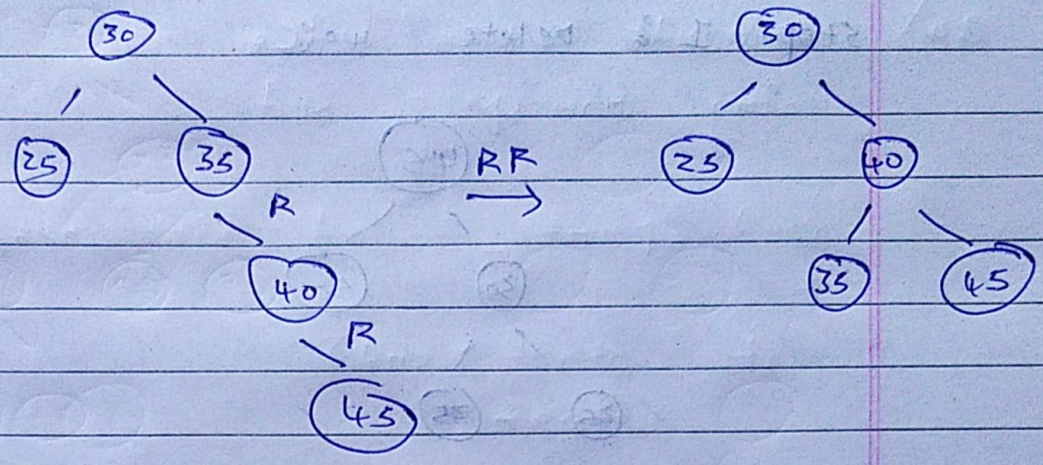
step 3 :



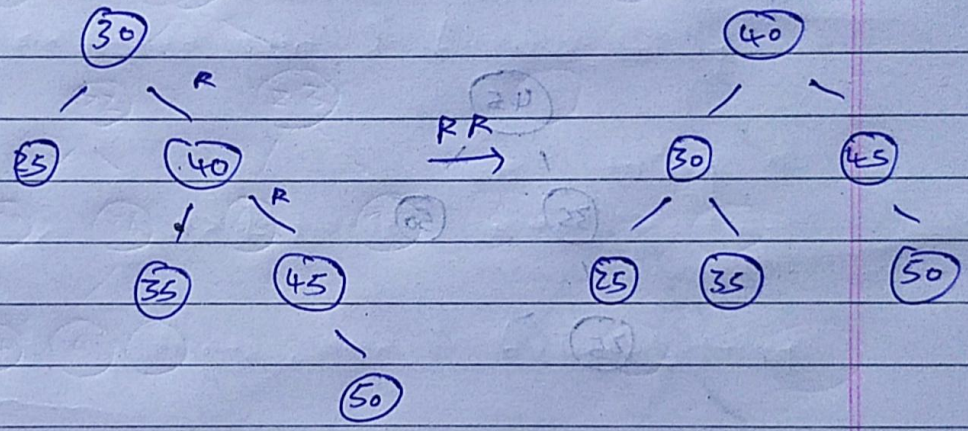
step 4 :



Step 5 :

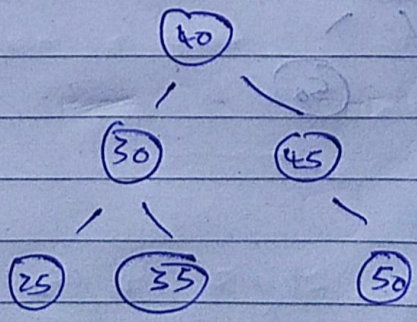


Step 6 :

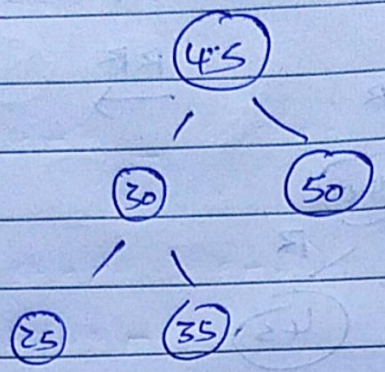


(ii) Delete 40, 30, 45

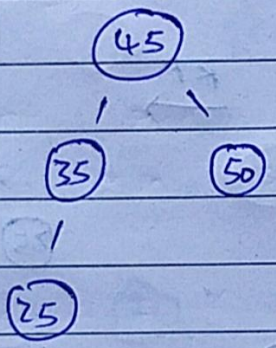
AVL Tree :



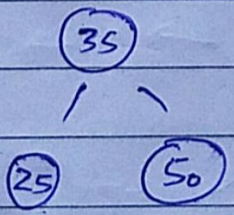
step 1 : Delete 40 :



step 2 : Delete 30

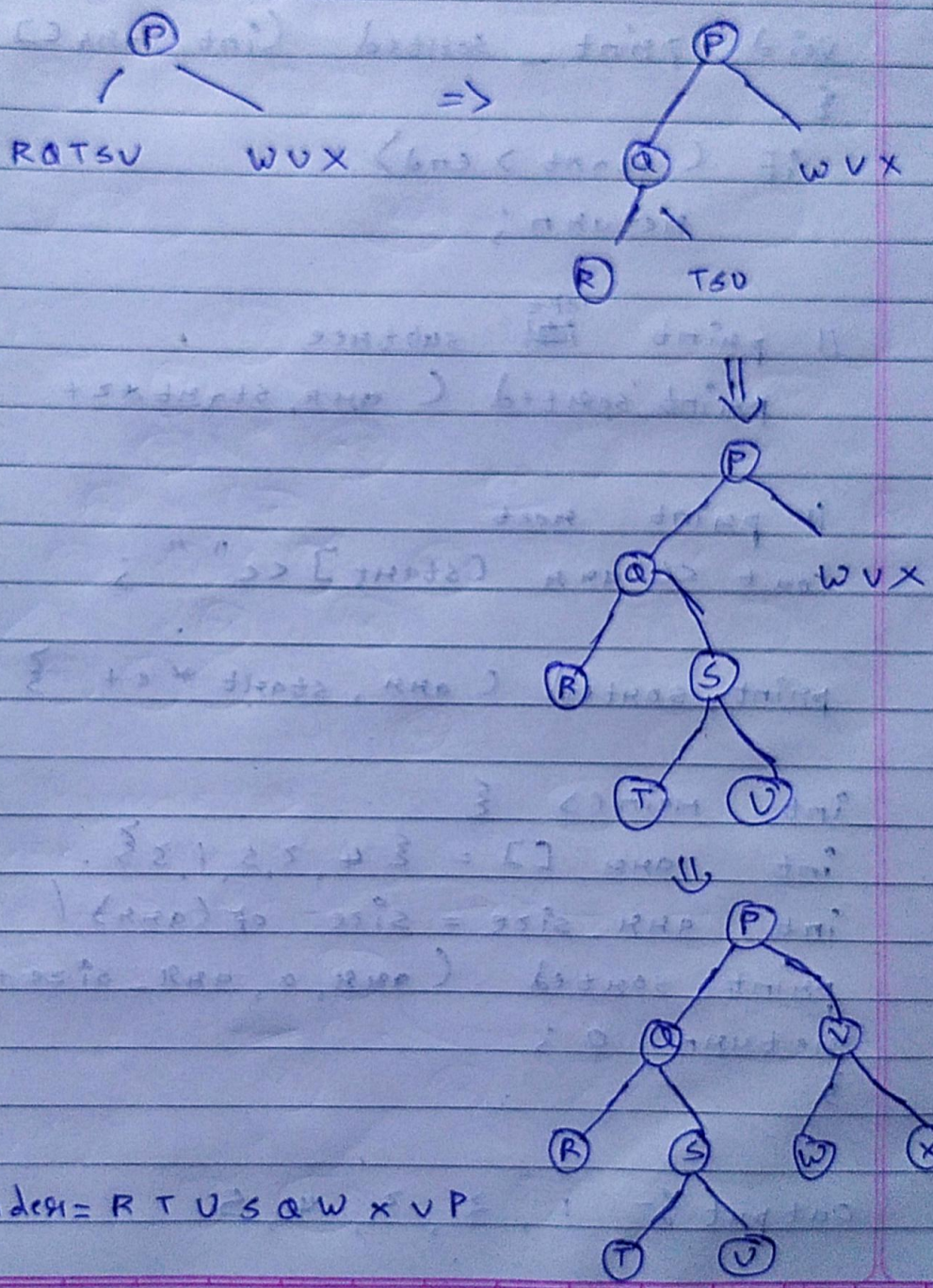


step 3 : Delete 45



Q7) Pre order = R Q R S T U V W X

In order = R Q T S U P W V X



Post order = R T U S Q W X V P

a4)
```
#include <bits/stdc++.h>
using namespace std;

void print_sorted (int arr[], int
{
  if (start > end)
      return;

// print left subtree
  print_sorted ( arr, start*2+

// print root
cout << arr [start] << " ";

print_sorted ( arr, start * c+ }

int main() {
int arr [] = { 4, 2, 5, 1, 3}
int arr_size = size of (arr) /
print_sorted (arr, 0, arr_size+char();
return 0;
}
```

Output :- 1, 2, 3, 4, 5