[This question paper contains 6 printed pages.]

Your Roll No.....

Sr. No. of Question Paper : 6504 **HC**

Unique Paper Code : 32341301

Name of the Paper : Data Structures

Name of the Course : **B.Sc. (H) Computer Sc.**

Semester : III

Duration : 3 Hours Maximum Marks : 75

Instructions for Candidates

1. Write your Roll No. on the top immediately on receipt of this question paper.
2. Attempt any **four** questions out of the remaining **Q.2-Q.7.**
3. Parts of a question must be answered together.

1. (a) Write enqueue and dequeue functions for a queue to be implemented through a circular singly linked list. (5)

   (b) Given the following code. Write its recursive function. (5)

P.T.O.

```
Void f(int n)
{
For (i=1;i<=n;i++)
{
If(i%2==0)
Cout<<i*i*i;
}
}
```

(c) Evaluate the following postfix expression using a stack:

4 10 5 + * 15 3 / -

Show the contents of the stack after every step. (5)

(d) Sort the following set of elements using selection sort. Show the content of array after every pass.

34, 56, 12, 8, 92, 9, 44, 23. (5)

(e) A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below. (2+3=5)

0	
1	11
2	32
3	63
4	71
5	85
6	52
7	
8	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table? Justify your answer.

(i) 85, 11, 63, 71, 32, 52

(ii) 63, 11, 32, 71, 52, 85

(iii) 85, 63, 11, 32, 71, 52

(iv) 11, 85, 52, 32, 63, 71

(f) Some search operations are to be performed on a sorted data stored in an array. However, it is known that the keys to be searched are all present in the initial few positions. Which search technique would you use? Justify your answer. (2+3=5)

P.T.O.

(g) Construct a binary tree whose following traversals are given : (5)

Inorder: x y z a p q r

Preorder: a y x z q p r

2. (a) Write a recursive function to display a single linked list of integers in reverse order. (4)

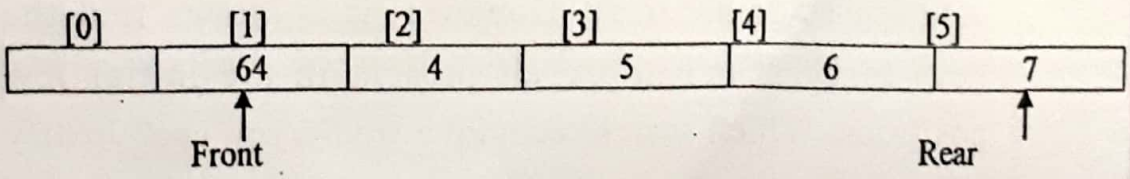
(b) Write a member function to delete the element at i^{th} position in a doubly linked list. The position i is passed as a parameter to this function. (6)

3. (a) Write a function to reverse the order of elements in stack using two additional stacks. (4)

(b) Compare and contrast the behavior of bubble and insertion sort on the following set of values.

1, 2, 3, 4, 5, 6. (4)

4. (a) Given a queue implemented using array of size 6. Show the queue and the front & rear values after performing each of the following operations.



enqueue(14), enqueue(56), dequeue(), dequeue(). (4)

(b) What is hashing? Explain any two hashing functions. Explain linear probing method of collision resolution with an example. (2+2+2=6)

5. (a) What are Self organizing lists? Compare the following two methods used to self organize lists.

(i) Move to Front

(ii) Transpose (2+3=5)

(b) Give the formula and calculate the address of the element A[3][2] of the 2D array defined as A[5]5], if the elements are stored in

(i) Row major order.

(ii) Column major order.

The beginning address of array is 400. Every element requires 4 bytes of storage. (4) (5)

6. (a) Create a binary search tree using the following values. 12, 45, 13, 67, 10, 34.

Using the above tree perform the following operations

(i) Delete 12 using delete by merging.

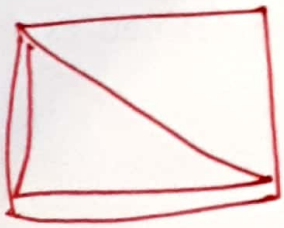
(ii) Delete 45 using delete by copying. (6)

(b) Write a function to calculate the number of leaves in a binary tree. (4)

7. (a) Insert the following values in B tree of order 5.

45, 12, 34, 78, 90, 22, 88, 96, 40, 82, 55, 100. (6)

(b) Define a class to implement a Lower Triangular matrix as a 1D array. Write a member function to store and retrieve its elements. (4)



1(c) Enqueue 2½ (1) (3234130)

Dequeue 2½

b) Void myfunc (int n)

{ if (n==0) return;

else

{ myfunc (n-1);

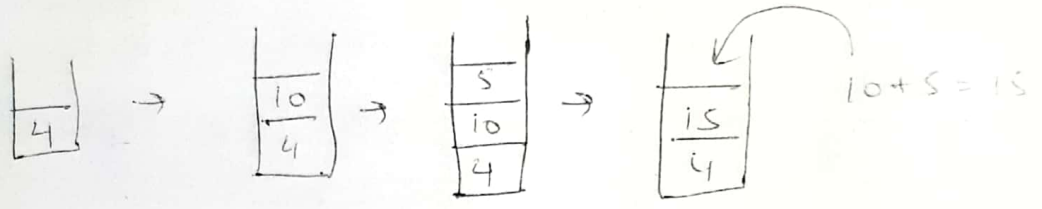
if ((n%2)==0)

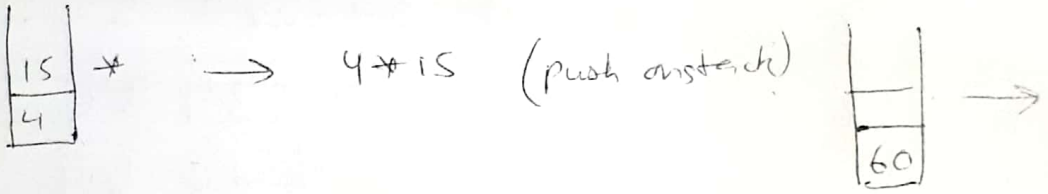
cout << n << " ";

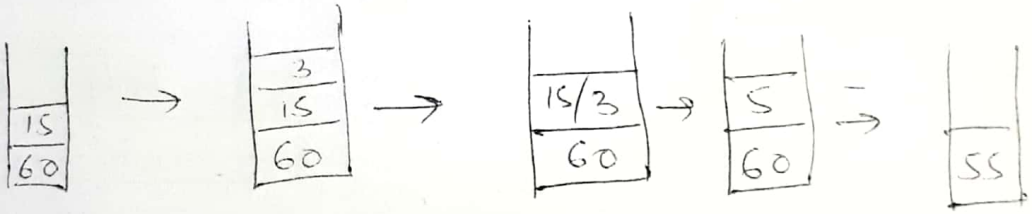
} }

5 marks

c)







5 marks

If only final answer is given give 1 mo

d)

d) 34 56 12 (8) 92 9 44 23

Step 1 find smallest element and swap with 1st element

8 56 12 34 92 (9) 44 23

Step 2: find smallest in remaining (n-1) element & ~~swap~~ swap with 2nd el

8 9 12 34 92 56 44 23

Similarly

8 9 12 34 92 56 44 23

8 9 12 23 92 56 44 34

8 9 12 23 34 56 44 92

8 9 12 23 34 44 56 92

8 9 12 23 34 44 56 92

5 marks

e) 2 hirdis correct 2 marks

Justification 3 marks

(f) If few means ≤ 5 then linear

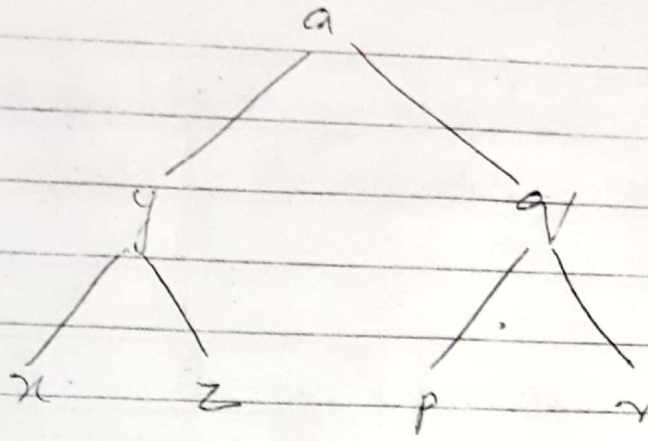
otherwise binary is also acceptable

Give marks according to justification given

both linear & binary may be correct

according to justification

g)



5 marks

2a)

```
void printReverse ( Node * head)
{   if (head == null) return;
    printReverse (head -> next);
    cout << head -> info;
}
```

4 marks.

b) 6 marks for function.

Ans 3. a)

reverse order of elements in stack using two additional stack.

```
int main()
{
    stack<int> s, s1, s2;
    s.push(i) // insert elements in s;
    while (!s.empty())
        s1.push(s.pop());
    while (!s1.empty())
        s2.push(s1.pop());
    while (!s2.empty())
        s.push(s2.pop());
```

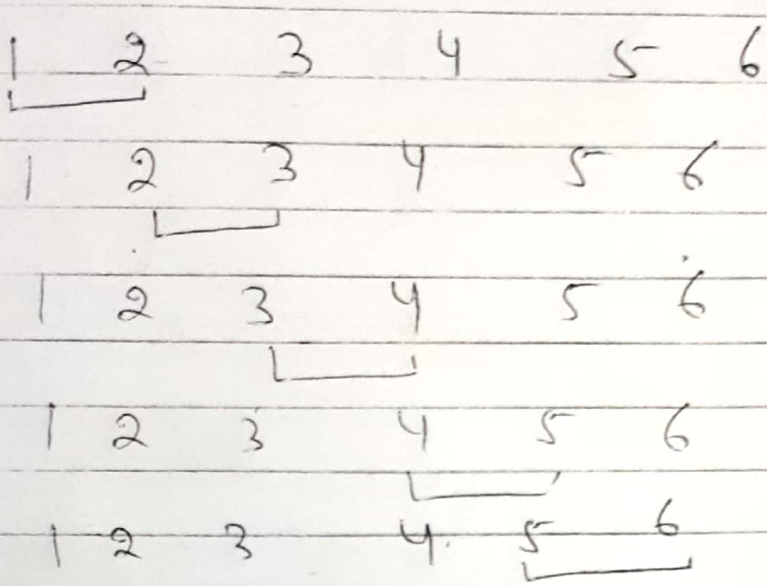
Now all print all elements of stack s (they will be in reverse order)

b) sequence is

1 2 3 4 5 6

bubble sort :

Step 1:

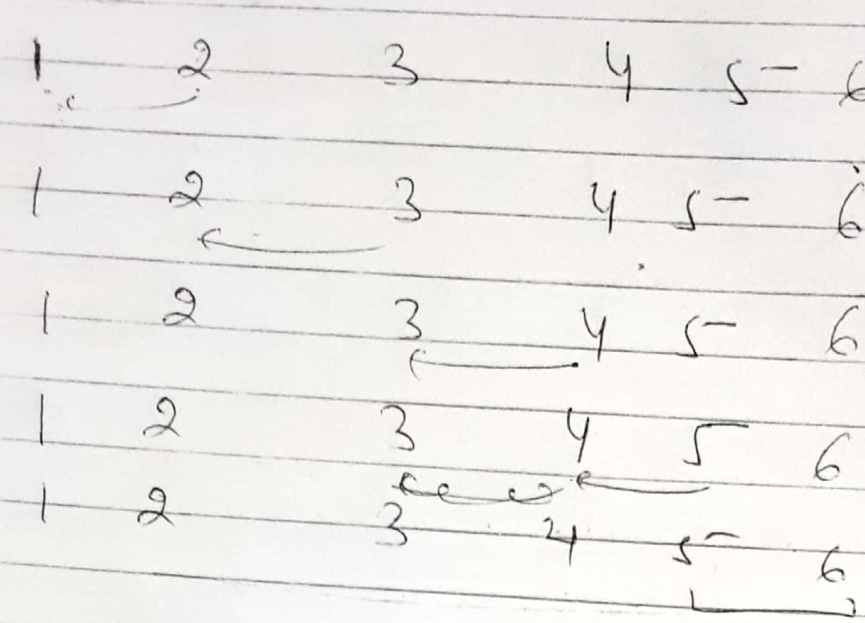


In bubble sort in first step it will make all the comparisons whether swapping has been done or not.

We can stop the loop after first step if there is no swapping in this step. So there will be maximum five comparisons.

(6)

Insertion sort:-



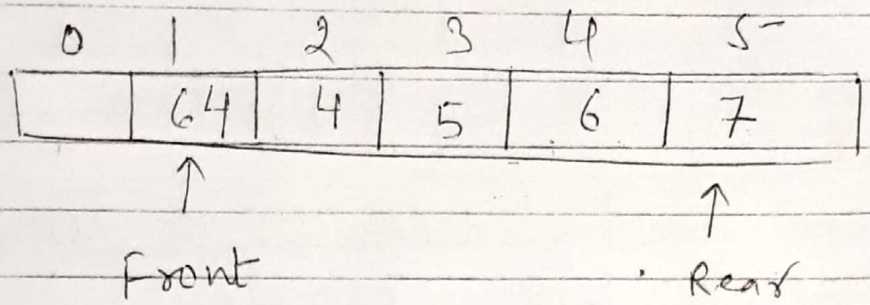
There are maximum of 4 comparisons in this case

In general insertion sort is better than bubble sort as you can stop inner loop earlier after you found current position of the element.

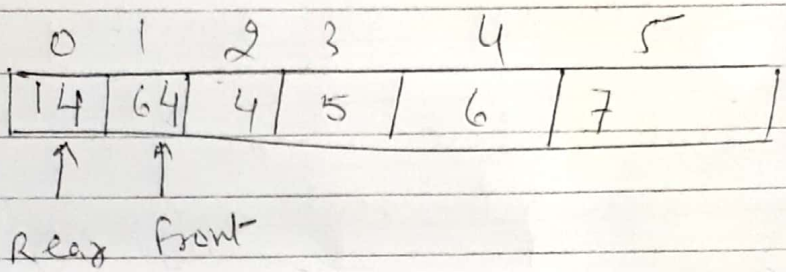
6 marks (correct)

7

Any 4 a)

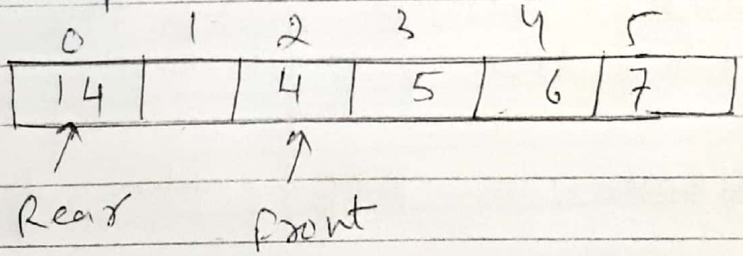


(i) enqueue (14)

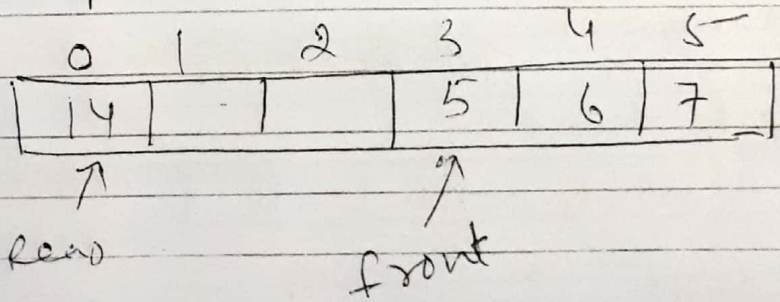


(ii) enqueue (56) Overflow error

(iii) dequeue



(iv) dequeue



4 b) defⁿ of hashing

any of two hashing functions

linear probing method of collision resolution with example

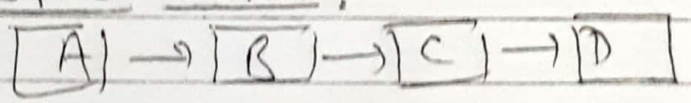
2+2+2 = 6

5. a) Efficiency of searching in a linked list can be improved by dynamically organizing the list 2 marks

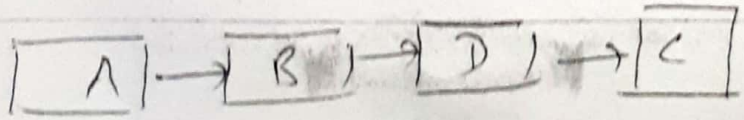
Move to front: After the desired element is located put it at the beginning of the list

Transpose: After the desired element is located swap it with its predecessor. 1½ marks

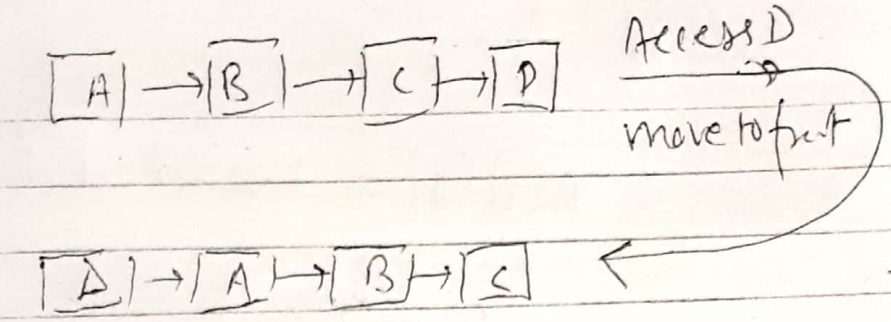
transpose method:



1½ marks Access D



9



5 b.) A[5][5]

base address = 400

Address of A[3][2]

i) row major order:

address of A[i][j] = Base addres + (i x n + j) * s

n = no. of columns

s = size of each element

address of A[3][2] = 400 + (3 x 5 + 2) x 4

= 400 + 17 x 4

= 468

Column major :

Address of $A[i][j]$ = base add. $+ (i + j \times m) * s$

m = no. of rows

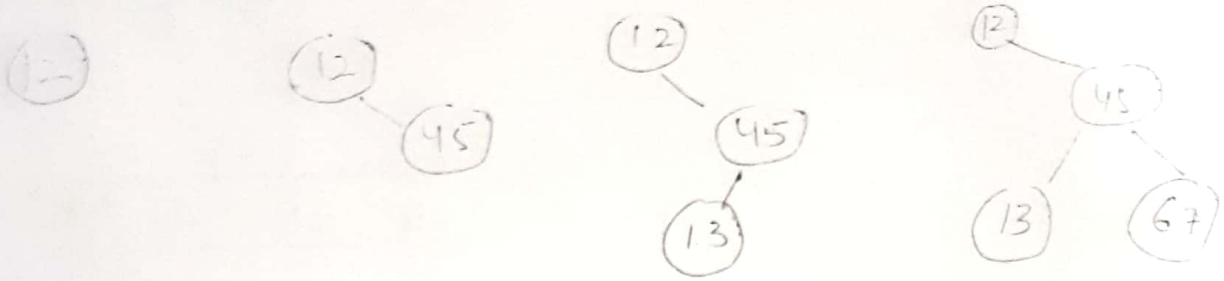
s = size

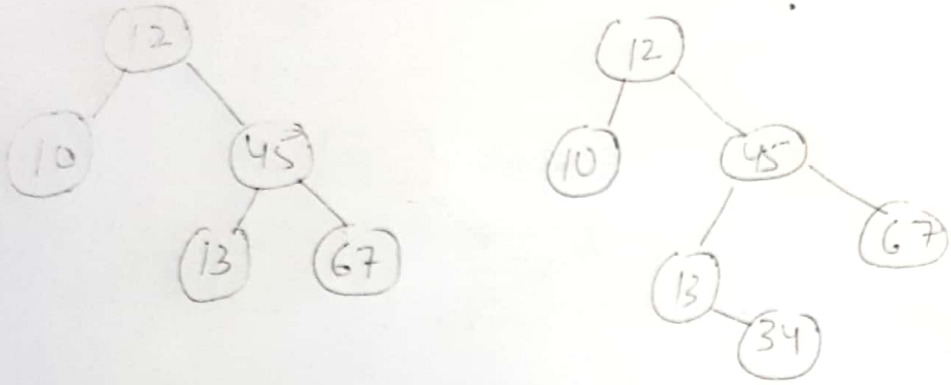
Add. of $A[3][2]$ = $400 + (3 + 2 \times 5) * 4$

$= 400 + 13 \times 4$

$= 452$

(4)



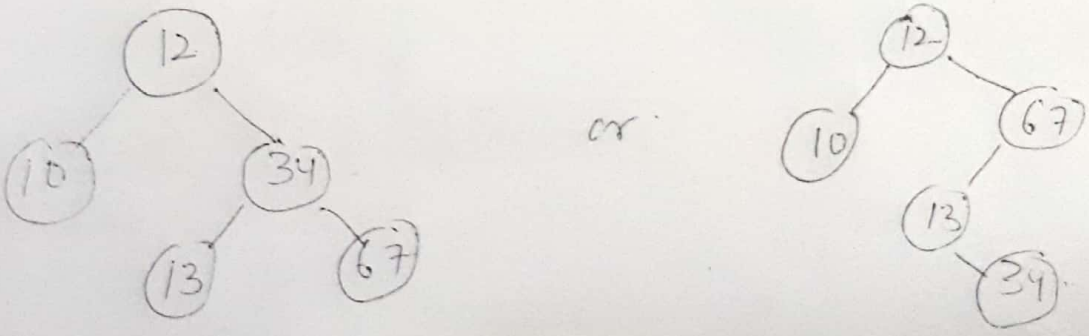


(i) Deleting 12 using delete by merging

Inordu successor

Inovdu Predecessor

(ii) Deleting 45 using delete by copying



Inorder successor

Inorder predecessor

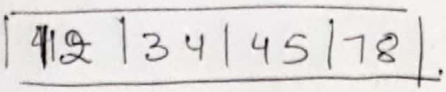
if deleted from the tree created in (i) ✓

(b)

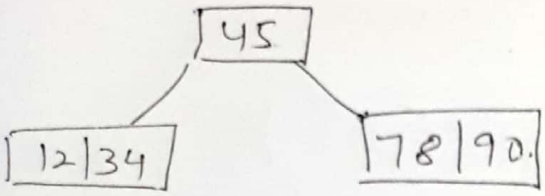
```
int count_leaf (tnode x t)
{
    if (t == NULL) return 0;
    else
        if (t->lchild == 0 && t->rchild == 0)
            return 1;
        else
            return (count_leaf (t->lchild) +
                    count_leaf (t->rchild));
}
```

Recursive or Iterative both are correct.

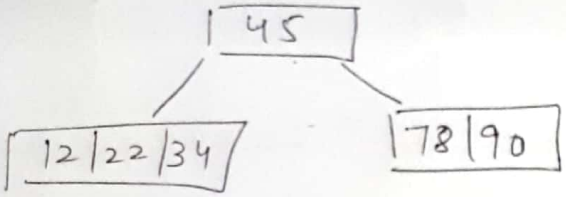
Que 7 Inserting 45, 12, 34, 78



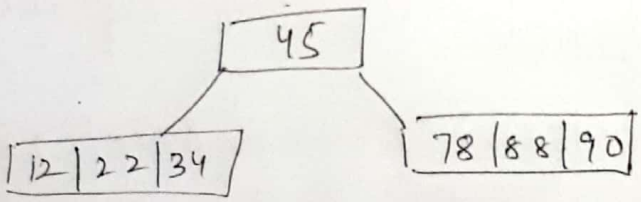
Inserting 90.



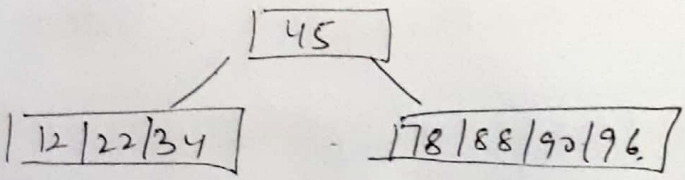
Inserting 22.



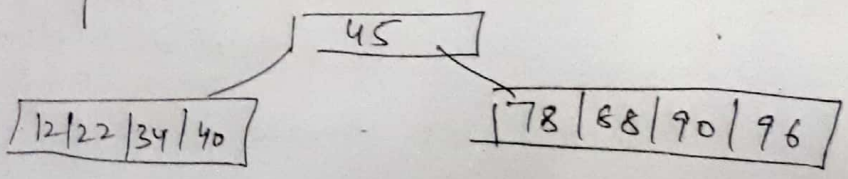
Inserting 88



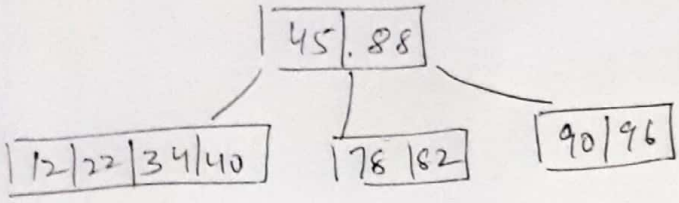
Inserting 96



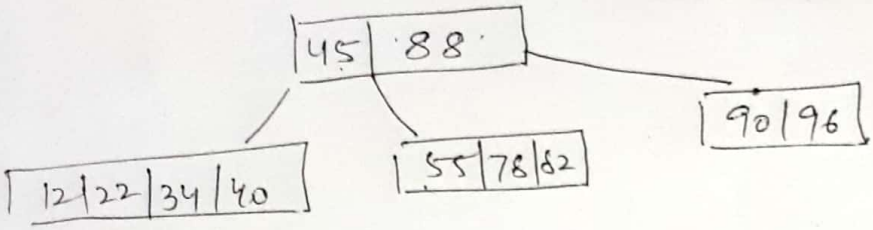
Inserting 40



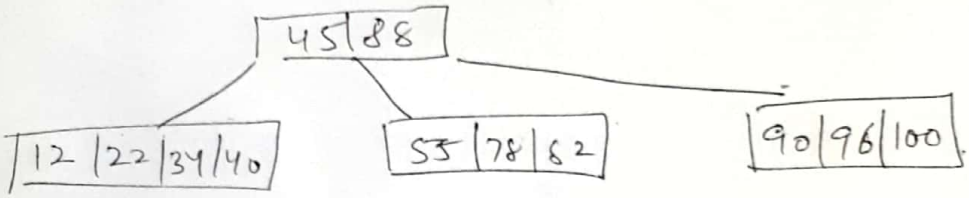
Inserting 82



Inserting 55



Inserting 100.



(b)

```
class LMatrix
{ int *a;
  int n;
public:
    LMatrix (int n)
    { if (n<0) throw "Invalid Size";
      a = new int [n]
      this->n = n;
    }

void store (int i, int j, int x)
{ if ( i<=0 || i>n || j<=0 || j>n)
                throw " Invalid index";
  if (j>i && x!=0) throw "Invalid value";
```

```
if (i<=j)
{ int K=(i-1)*i/2 + j-1;
  a[K] = x;
}
}
int Retrieve (int i, int j)
{ if (i<=0 || i>n || j<=0 || j>n)
    throw "Invalid Index";
  if (j>i) return 0;
  else { int K = (i-1)*i/2 + j-1;
         return a[K];
       }
}
```

```
int dequeue(Queue *q)
{ if (q->front==NULL)
  { printf("Empty");
  } int value;
  if(q->front==q->rear)
  { value = q->front->data;
  free(q->front);
  q->front = NULL;
  q->rear = NULL;
}
```

```
else
{
struct Node *temp = q->front;
  value = temp->data;
q->front = q->front->link;
q->rear->link = q->front;
; free(temp);
}
return value;
}
```

```
struct Node
{ int data;
  struct Node *link;
};

struct Queue
{
  struct Node *front, *rear;
};
```

```
Void enQueue(Queue *q, int value)
{ struct Node *temp = new Node;
  temp -> data = value;
 if (q -> front == NULL)
     q -> front = temp;
 else
     q -> rear -> link = temp;

   q -> rear = temp;
  q -> rear -> link = q -> front;
```
