Q-1 How to find the mid node in LinkedList?

→

```
class Node {
    public:
        int data;
        Node* next;
};

int getlen(class Node* head)
{
    int len=0;
    class Node* temp = head;
     while (temp) {
          len++;
          temp = temp -> next;
     }
    return len;
}
void printMiddle( class Node* head)
{
     if(head) {
          int len = getLen(head);
          class Node* temp= head;
          int midInd = len/2;
          while( midInd --) {
                    temp = temp -> next;
          }
{    cout << temp -> data << endl;
```

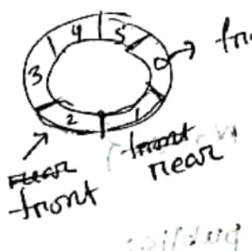
NB.
because:

2→4→5→6

here, 5 is middle node

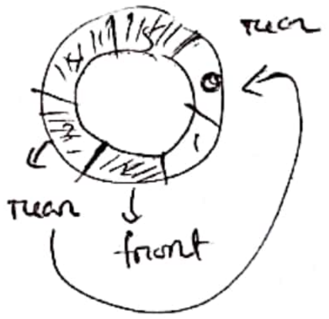
Qus-02

Circular Queue:

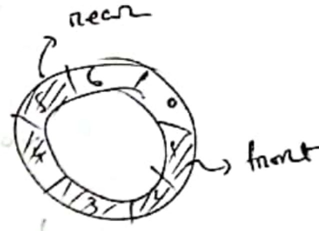


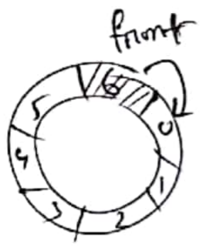
front = = 0
rear = 6-1=5

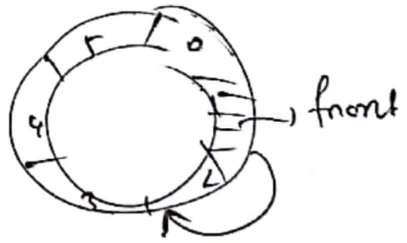
```
void enQueue (int value) {

    if ((front==0 && rear== size-1) ||
            rear==(front-1))

        cout << " Queue is full"<< endl;

    else if ( front==-1) {      // insert 1st element

        front= rear=0;
        arr [rear]= value;
    }

    else if ( rear==size-1 && front !=0) {

        rear=0;
        arr [rear] = value;
    }

    else {
        rear++;
        arr [rear]= value;
    }
}
```



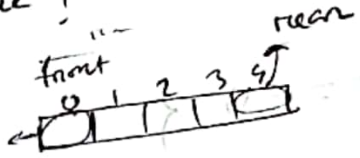
```
int   dequeue() {

    if ( front == -1) {
        cout << "  Queue is  Empty " <<endl;
    }

    int data = arr [front];

    arr [front] = -1;   // initialize as -1

    if ( front == rear) {   //. one element

        front = -1;
        rear = -1;
    }

    else if ( front == size -1) {
        front = 0;
    }

    else
        front++;

    return   data;
```







Q-3 How to reverse a queue?

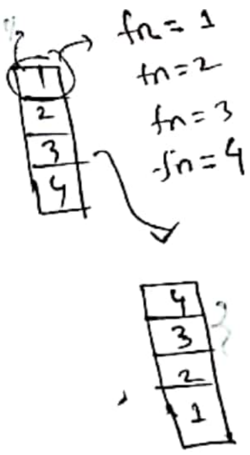


→

```
void reverseQueue ( queue <int>& Queue)

{

    stack < int > stack;

    while ( ! Queue . empty()) {

        stack. push ( Queue. front());

        Queue. pop();

    }

    while ( ! stack. empty()) {

        Queue. push ( stack. pop());

        stack. pop();

    }

}
```

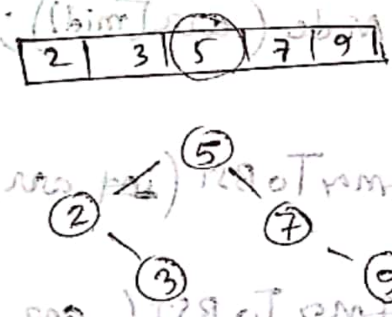
using recursion

```
void  reverse Queue( queue<int> &q)
{
        if ( q.size() == 0)
             return;

        int fr = q.front();

        q.pop();

        reverse Queue(q);

        q.push(fr);

}
```



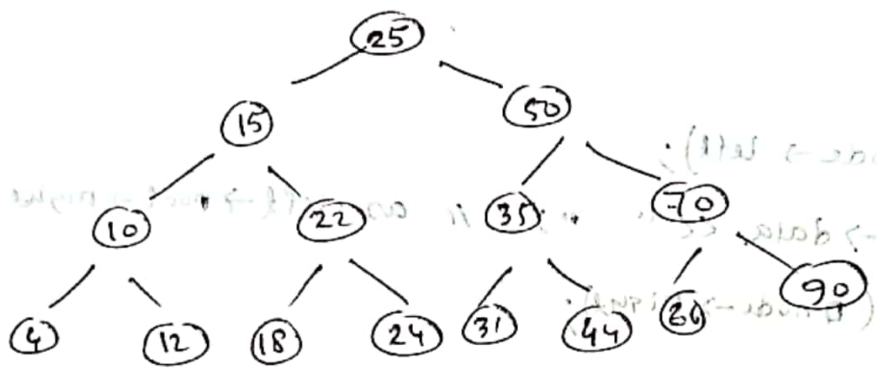
Q-4 How to creat BST from an Sorted Array?

→



```
class  Tnode {
        public:
            int data;
          Tnode*  left;
          Tnode*  right;
}

Tnode*    sortedArrayToBST ( int  arr[], int start, int end)

}
    if ( start > end)
        return NULL;

    int   mid = ( start + end) / 2;

    Tnode*   root = new Node( arr[mid]);

    root -> left  = sortedArrayToBST (~~int~~ arr, start, mid-1);

    root -> right  =  sortedArrayToBST ( arr, mid +1, end);


    return  root;

{
```

Q-un-05 Tree Output:



In-Order Traversal: 4 10 12 15 18 22 24 25 31 35 44
50 66 70 90

Pre-order Traversal: 25 15 10 4 12 22 18 24 50
(Root → left → right) 35 31 44 70 66 90

Post-order Traversal: 4 12 10 18 24 22 15
(left → right → Root) 31 44 35 66 90 70 50 25

```
void  printInorder( struct Node* node)

  {   if (node == NULL)
             return;
      printInorder(node -> left);
       cout << node -> data << " ";  // as left -> root -> right
        printInorder(node -> right);

  }

void  preorder (struct Node* node)
  {   if  (node == NULL)
              return;
       cout << node -> data << " ";   // as  root -> left -> right
       preorder( node -> left);
       preorder  (node -> right);
  }

void  postorder( struct Node* node)
  {    if  (node == NULL)
              return;
     postorder  ( node -> ~~right~~ left);
     postorder( node -> right);
     cout << node -> data << " ";
  }
```