* **Q.1415111**: Search an element in a Linked List (Recursive)

import java.util.\*;

class Node

{

int data;

Node next;

Node(int d)

{

data = d;

next = null;

}

}

// Linked list class

class TestClass

{

Node head;

public void push(int new\_data)

{

//Allocate new node and putting data

Node new\_node = new Node(new\_data);

//Make next of new node as head

new\_node.next = head;

head = new\_node;

}

public boolean search(Node head, int x)

{

if (head == null)

return false;

if (head.data == x)

return true;

return search(head.next, x);

}

public static void main(String args[])

{

int n,p;

TestClass llist = new TestClass();

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

for(int i=0;i<n;i++)

{

int r=sc.nextInt();

llist.push(r);

}

p=sc.nextInt();

if (llist.search(llist.head, p))

System.out.println("Yes");

else

System.out.println("No");

}

}

Swapping of Nodes –SLL

import java.util.\*;

class Node

{

int data;

Node next;

Node(int d)

{

data = d;

next = null;

}

}

class TestClass

{

Node head; // head of list

/\* Function to swap Nodes x and y in linked list by

changing links \*/

public void swapNodes(int x, int y)

{

// Nothing to do if x and y are same

if (x == y) return;

// Search for x (keep track of prevX and CurrX)

Node prevX = null, currX = head;

while (currX != null && currX.data != x)

{

prevX = currX;

currX = currX.next;

}

// Search for y (keep track of prevY and currY)

Node prevY = null, currY = head;

while (currY != null && currY.data != y)

{

prevY = currY;

currY = currY.next;

}

// If either x or y is not present, nothing to do

if (currX == null || currY == null)

return;

// If x is not head of linked list

if (prevX != null)

prevX.next = currY;

else //make y the new head

head = currY;

// If y is not head of linked list

if (prevY != null)

prevY.next = currX;

else // make x the new head

head = currX;

// Swap next pointers

Node temp = currX.next;

currX.next = currY.next;

currY.next = temp;

}

/\* Function to add Node at beginning of list. \*/

public void push(int new\_data)

{

/\* 1. alloc the Node and put the data \*/

Node new\_Node = new Node(new\_data);

/\* 2. Make next of new Node as head \*/

new\_Node.next = head;

/\* 3. Move the head to point to new Node \*/

head = new\_Node;

}

/\* This function prints contents of linked list starting

from the given Node \*/

public void printList()

{

Node tNode = head;

while (tNode != null)

{

System.out.print("-->");

System.out.print(tNode.data);

tNode = tNode.next;

}

}

public static void main(String[] args)

{

int n,p,q;

TestClass llist = new TestClass();

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

for(int i=0;i<n;i++)

{

int r=sc.nextInt();

llist.push(r);

}

p=sc.nextInt();

q=sc.nextInt();

System.out.println("Linked list before Swapping");

llist.printList();

llist.swapNodes(p,q);

System.out.println("\nLinked list after Swapping");

llist.printList();

}

}