Loop detection

```
import java.util.*;class Node {int num;Node next;Node(int val) {num=val;next=null;} }class Main {static Node insertNode(Node head, int val) {Node newNode = new Node(val);if(head==null) {head=newNode;return head;}Node temp=head; while(temp.next!=null) temp=temp.next; temp.next=newNode; return head;}static void display(Node head) { Node temp = head; while(temp.next!=null) { System.out.print(temp.num+"->"); temp=temp.next;} System.out.println(temp.num+"->"+"NULL");}static void createCycle(Node head,int a,int b) { int cnta = 0,cntb = 0;Node p1 = head;Node p2 = head;while(cnta != a || cntb != b) {if(cnta != a) p1 = p1.next; ++cnta; if(cntb != b) p2 = p2.next; ++cntb; }p2.next = p1; }static boolean cycleDetect(Node head) { if(head == null) return false; Node fast = head; Node slow = head;while(fast.next != null && fast.next.next != null) { fast = fast.next.next;slow = slow.next;if(fast == slow) return true; }return false;}public static void main(String args[]) { Scanner sc = new Scanner(System.in);int n=sc.nextInt();Node head = null;for(int i=0;i<n;i++){int m=sc.nextInt(); head=insertNode(head,m);}display(head);int a=sc.nextInt(); createCycle(head,1,a);if(cycleDetect(head) == true) System.out.println("Cycle detected");elseSystem.out.println("Cycle not detected");}}
```

Stock span

```
import java.util.*;public class Main {static void calculate(int arr[], int n, int S[]){ Stack<Integer> st = new Stack<>();st.push(0);S[0] = 1;for (int i = 1; i < n; i++) { while (!st.isEmpty() && arr[st.peek()] <= arr[i])st.pop();S[i] = (st.isEmpty()) ? (i + 1) : (i - st.peek());st.push(i);}}static void printArray(int arr[]){ System.out.print(Arrays.toString(arr));}public static void main(String[] args){ Scanner sc=new Scanner(System.in);int n=sc.nextInt();int arr[]=new int [n]; for(int i=0;i<n;i++)arr[i]=sc.nextInt();int S[] = new int[n];calculate(arr, n, S); printArray(S);}}
```

Maximum sliding window

```
import java.util.*;public class Main {public static int[] maxsliding( int[]in,int w){ int[] max_left = new int[in.length];int[] max_right = new int[in.length]; max_left[0]=in[0];max_right[in.length-1]=in[in.length-1];for (int i=1;i<in.length;i++){max_left[i]=(i%w==0)?in[i]:Math.max(max_left[i-1],in[i]); final int j=in.length-i-1;
max_right[j]=(j%w==0)?in[j]:Math.max(max_right[j+1],in[j]);}final int[] sliding_max=new int[in.length-w+1]; for(int i=0,j=0;i+w<=in.length;i++){ sliding_max[j++]=Math.max(max_right[i],max_left[i+w-1]);}return sliding_max; }public static void main(String[] args) throws Exception{Scanner sc=new Scanner(System.in);int n=sc.nextInt();int k=sc.nextInt();int a[]=new int[n];for(int i=0;i<n;i++){a[i]=sc.nextInt();}int ans[]=maxsliding(a,k); for(int i=0;i<ans.length;i++)System.out.print(ans[i]+" ");}}
```

Heap sort

```
import java.util.*;public class Main {public static void sort(int arr[]){int N=arr.length;
for(int i=N/2-1; i>=0; i--)heapify(arr,N,i);for(int i=N-1; i>0; i--) {int temp=arr[0];arr[0]=arr[i];
arr[i]=temp;heapify(arr,i,0);}}static void heapify(int arr[], int N, int i){int largest=i; int l=2*i+1; int r=2*i+2; if(l<N && arr[l]>arr[largest])largest=l;if(r<N && arr[r]>arr[largest])
largest=r;if(largest!=i) {int swap=arr[i];arr[i]=arr[largest];arr[largest]=swap;
heapify(arr,N,largest);}}public static void main(String args[]){ Scanner s=new Scanner(System.in);int n=s.nextInt();int arr[] = new int[n]; for(int i=0; i<n; i++)arr[i]=s.nextInt();sort(arr);System.out.println("Sorted array is");for(int i=0; i<n; i++)
System.out.print(arr[i] + " ");System.out.println();}}
```

Hash map

```
import java.util.*; import java.util.stream.*; class Main {public static <K, V> Map<K, V> convertToTreeMap(Map<K, V> hashMap) { Map<K, V>treeMap = hashMap.entrySet().stream().collect(Collectors.toMap(Map.Entry::getKey,Map.Entry::getValue, (oldValue, newValue) -> newValue, TreeMap::new)); return treeMap; } public static void main(String args[]) {Map<String, String> hashMap = new HashMap<>(); Scanner s=new Scanner(System.in);int n=s.nextInt();for(int i=0; i<n; i++)hashMap.put(s.next(),s.next()); System.out.println("HashMap: " + hashMap); Map<String, String> treeMap = convertToTreeMap(hashMap); System.out.println("TreeMap: " + treeMap); } }
2. public static <K, V> Map<K, V> convertToTreeMap(Map<K, V> hashMap) { Map<K, V> treeMap = new TreeMap<>(); treeMap.putAll(hashMap); return treeMap; }

3. public static Map<Integer, String> convertToTreeMap(Map<String, String> hashMap){ Map<Integer, String> treeMap = new TreeMap<>(); for (Map.Entry<String, String> e : hashMap.entrySet())treeMap.put(Integer.parseInt(e.getKey()), e.getValue()); return treeMap; }
```

Subset sum problem

```
import java.util.Scanner;class Main {static boolean isSubsetSum(int set[], int n, int sum){if (sum == 0)return true;if (n == 0)return false;if (set[n - 1] > sum)
return isSubsetSum(set, n - 1, sum);return isSubsetSum(set, n - 1, sum) || isSubsetSum(set, n - 1, sum - set[n - 1]);}public static void main(String args[]){
Scanner sc = new Scanner(System.in);int n =sc.nextInt();int sum = sc.nextInt();
int set[]= new int[n]; for(int i=0;i<n;i++)set[i]=sc.nextInt();if (isSubsetSum(set, n, sum) == true)System.out.println("yes");else System.out.println("no");}}
```

Celebrity problem

```
import java.util.*;public class Main {public static int celebritySolution(int[][] mat) {Stack<Integer> stk = new Stack<>();for(int i=0;i<mat.length;i++){stk.push(i);}while(stk.size() > 1) {int col = stk.pop();int row = stk.pop();if(mat[row][col] == 1) {stk.push(col);} else {stk.push(row); }}int x = stk.pop();for(int j=0;j<mat.length;j++) {if(j == x) continue;if(mat[x][j] == 1) {return -1;}} for(int i=0;i<mat.length;i++) {if(i == x) continue;if(mat[i][x] == 0) {return -1;}}return x;}public static void main(String[] args){Scanner sc=new Scanner(System.in);int n=sc.nextInt();int matrix[][]=new int[n][n];for(int i=0;i<n;i++)for(int j=0;j<n;j++)matrix[i][j]=sc.nextInt();int res = celebritySolution(matrix);if(res == -1) System.out.println("There is no celebrity in the party");else System.out.println(res + " is the celebrity in the party");}}
```

Stack Permutation

```
import java.util.*;class Main {static Boolean check(int ip[], int op[], int n){Stack<Integer> s = new Stack<Integer>();int j = 0;for (int i = 0; i < n; i++) {s.push(ip[i]);while (!s.isEmpty() && s.peek() == op[j]) {s.pop();
j++;}}if (s.isEmpty())return true;return false;}public static void main(String args[]){Scanner sc=new Scanner (System.in);int n=sc.nextInt();int input[]=new int[n];int output[]=new int[n];for(int i=0;i<n;i++)input[i]=sc.nextInt();for(int i=0;i<n;i++)
output[i]=sc.nextInt();if (check(input, output, n))
System.out.println("Yes");else System.out.println("Not Possible");}}
```

Distributing items when a person cannot take more than two items of same type

```
import java.util.*;public class Main {static boolean checkCount(int []arr, int n, int k){int count;for (int i = 0; i < n; i++){count = 0;for (int j = 0; j < n; j++){if (arr[j] == arr[i])count++;if (count > 2 * k)return false;}}return true;}static public void main (String[] args){Scanner s=new Scanner(System.in);int n=s.nextInt(); int k = s.nextInt();int []arr =new int[n];for(int i=0; i<n; i++) arr[i]=s.nextInt(); if(checkCount(arr, n, k)) System.out.println("Yes"); else System.out.println("No");}}
```

Longest biotonic subsequence

```
import java.util.*;public class Main{public static int lbs(int arr[],int n){ int lis[] = new int[n]; for(int i=0; i<n; i++) lis[i] = 1; for(int i=1; i<n; i++) for(int j=0; j<i; j++) if(arr[i]>arr[j] && lis[i]<lis[j]+1) lis[i]=lis[j]+1; int lds[] = new int [n]; for(int i=0; i<n; i++) lds[i]=1; for(int i=n-2; i>=0; i--) for(int j=n-1; j>i; j--) if (arr[i]>arr[j] && lds[i]<lds[j]+1) lds[i]=lds[j]+1; int max=lis[0]+lds[0]-1; for(int i=1; i<n; i++) if(lis[i]+lds[i]-1>max) max=lis[i]+lds[i]-1; return max; }public static void main(String args[]){Scanner s=new Scanner(System.in);int n=s.nextInt();int a[]=new int[n];for(int i=0; i<n; i++)a[i]=s.nextInt();System.out.print(lbs(a,n));}}
```

Longest common subsequence

```
import java.util.Scanner;public class Main {public static void main(String[] args) { Scanner sc = new Scanner(System.in);String str1 = sc.next();String str2 = sc.next(); Integer[][] dp = new Integer[str1.length()][str2.length()]; System.out.println(helper(0, 0, str1, str2, dp));}public static int helper(int i, int j, String str1, String str2, Integer[][] dp) {if (i == str1.length() || j == str2.length()) return 0; if (dp[i][j] != null)return dp[i][j];if (str1.charAt(i) == str2.charAt(j)) return 1 + helper(i + 1, j + 1, str1, str2, dp);int first = helper(i + 1, j, str1, str2, dp); int second = helper(i, j + 1, str1, str2, dp);int answer = Math.max(first, second); dp[i][j] = answer;return answer;}}
```

Longest palindromic subsequence

```
import java.util.Scanner;public class Main {public static void main(String args[]) {Scanner sc = new Scanner(System.in);String s = sc.next();Integer[][] dp = new Integer[s.length()][s.length()];System.out.println(helper(0, s.length() - 1, s, dp));}public static int helper(int start,int end,String s,Integer[][] dp) {if (start>end)return 0;if (start==end)return 1;if (dp[start][end]!=null)return dp[start][end];if (s.charAt(start)==s.charAt(end)) return dp[start][end]=2+helper(start+1,end-1,s,dp);
return dp[start][end]=Math.max(helper(start+1,end,s,dp),helper(start,end-1,s,dp));}}
```

Longest increasing subsequence

```
import java.util.Scanner;public class Main {public static int lis(int arr[],int n ){ int lis[]=new int[n]; lis[0] = 1;  for(int i=1; i<n; i++){ lis[i] = 1; for(int j=0; j<i; j++) if(arr[i]>arr[j] && lis[i]<lis[j]+1) lis[i]=lis[j]+1; } int m=Integer.MIN_VALUE; for(int i=0; i<n; i++)if(m<lis[i]) m=lis[i];return m;} public static void main(String[] args){ Scanner s=new Scanner(System.in);int n=s.nextInt();int a[]=new int[n];for(int i=0;i<n;i++)a[i]=s.nextInt();
System.out.print(lis(a,n));}}
```

Segregate even and odd nodes in a Linked List

```
import java.util.*;class Main {Node head;class Node {int data;Node next;Node(int d) {data = d;next = null;}}void segregateEvenOdd() {Node evenStart = null;Node evenEnd = null;Node oddStart = null;Node oddEnd = null;Node currentNode = head;while (currentNode != null) {int element = currentNode.data;if (element % 2 == 0) {if (evenStart == null) {evenStart = currentNode;evenEnd = evenStart;} else {evenEnd.next = currentNode;evenEnd = evenEnd.next;}} else {if (oddStart == null) {oddStart = currentNode;oddEnd = oddStart;} else {oddEnd.next = currentNode;oddEnd = oddEnd.next;}}currentNode = currentNode.next;}if (oddStart == null || evenStart == null) {return;}evenEnd.next = oddStart;oddEnd.next = null;head = evenStart;}void push(int new_data) {Node new_node = new Node(new_data);new_node.next = head;head = new_node;}void printList() {Node temp = head;while (temp != null) {System.out.print(temp.data + " ");temp = temp.next;}System.out.println();}public static void main(String args[]) {Main main = new Main();Scanner sc = new Scanner(System.in);int n = sc.nextInt();for (int i = 0; i < n; i++){int m = sc.nextInt(); main.push(m);}main.segregateEvenOdd();main.printList();}}
```

Sort without extra space

```
import java.util.*;class Main {public static int minIndex(Queue<Integer> list, int sortIndex) { int min_index = -1; int min_value = Integer.MAX_VALUE; int s = list.size(); for (int i = 0; i < s; i++) { int current = list.peek(); list.poll(); if (current <= min_value && i <= sortIndex) { min_index = i; min_value = current; } list.add(current); } return min_index; } public static void insertMinToRear(Queue<Integer> list, int min_index) {int min_value = 0; int s = list.size(); for (int i = 0; i < s; i++) { int current = list.peek(); list.poll(); if (i != min_index) list.add(current); else min_value = current;}list.add(min_value); }public static void sortQueue(Queue<Integer> list) { for(int i = 1; i <= list.size(); i++) { int min_index = minIndex(list, list.size() - i); insertMinToRear(list, min_index); }}public static void main (String[] args) {Queue<Integer> list = new LinkedList<Integer>(); Scanner sc = new Scanner(System.in); int n = sc.nextInt();for (int i = 0; i < n; i++) {int element = sc.nextInt();list.add(element);}sortQueue(list); System.out.println("Sorted queue:");while(!list.isEmpty()) { System.out.print(list.peek() + " "); list.poll();}}}
```

Minmum stack

```
import java.util.*;class Mystack {Stack<Integer> s;Stack<Integer> a;Mystack() {s = new Stack<Integer>();a = new Stack<Integer>();}void getMin() {if(a.isEmpty())System.out.println("Stack is Empty");else System.out.println("Minimum element : " + a.peek());}void peek(){if(s.isEmpty()) {System.out.println("Stack is Empty");return ;}int t=s.peek(); System.out.print("Top most element:" + t);}void pop() {int t = s.pop();if(s.isEmpty()) {System.out.println("Stack is Empty");return ;}else System.out.println("Removed element : " + t)if(t == a.peek())a.pop();}void push(int x) {if ( s.isEmpty()) { s.push(x);a.push(x);System.out.println(" Number Inserted: "+ x);return ;}else {s.push(x);System.out.println(" Number Inserted: " +x);}if ( x<= a.peek() ) a.push(x);}};public class Main {public static void main(String args[]) {Mystack s=new Mystack();Scanner sc = new Scanner(System.in);int n=sc.nextInt();for( int i=0;i<n;i++) {int m=sc.nextInt();s.push(m);}s.getMin();s.pop();s.getMin();s.pop();s.peek();}}
```