Cat 1 STS

1.Even after Odd Linked List.

```java
import java.util.*;

public class Solution {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] input = sc.nextLine().split(" ");
        List<Integer> odd = new ArrayList<>();
        List<Integer> even = new ArrayList<>();

        for (String num : input) {
            int val = Integer.parseInt(num);
            if (val == -1) break;
            if (val % 2 == 0) even.add(val);
            else odd.add(val);
        }

        odd.addAll(even);
        for (int val : odd) System.out.print(val + " ");
    }
}

```

```java
import java.util.*;
public class Solution{
    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();
        }
        for(int i=0;i<n;i++){
            if(arr[i]%2!=0){
                System.out.print(arr[i]+" ");
            }
        }
        for(int i=0;i<n;i++){
            if(arr[i]%2==0){
                System.out.print(arr[i]+" ");
            }
        }
    }
}

```

2. Loop detection

Main.java

```java
import java.util.*;

public class Main {
    public static boolean hasCycle(int[] arr, int pos) {
        if (pos >= arr.length || pos < -1) return false;
        if (pos != -1) arr[arr.length - 1] = pos;
        Set<Integer> visited = new HashSet<>();
        for (int i = 0; i < arr.length; i++) {
            if (visited.contains(arr[i])) return true;
            visited.add(arr[i]);
        }
        return false;
    }

    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();
        System.out.println(hasCycle(arr, sc.nextInt()) ? "Loop detected" : "No Loop detected");
    }
}

```

3. Sort the Bitonic DLL:

4.Merge sort DLL:

5.Sort the Queue without extra space

Main.java

```java
//Selection sort
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        for(int i=0;i<n;i++) a[i]=sc.nextInt();
        for(int i=0;i<n;i++){
            int index=i; //store least elements
            for(int j=i+1;j<n;j++){
                if(a[j]<a[index]){
                    index=j;
                }
            }
            int temp=a[i];
            a[i]=a[index];
            a[index]=temp;
        }
        for(int i=0;i<n;i++) System.out.print(a[i]+" ");
    }
}
```

```java
//Selection sort
import java.util.*;
public class Solution{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        for(int i=0;i<n;i++) a[i]=sc.nextInt();
        for(int i=0;i<n;i++){
            int index=i; //store least elements
            for(int j=i+1;j<n;j++){
                if(a[j]<a[index]){
                    index=j;
                }
            }
            int temp=a[i];
            a[i]=a[index];
            a[index]=temp;
        }
        System.out.print("[");
        for(int i=0;i<n;i++) {
            System.out.print(a[i]);
            if(i < n-1) {
                System.out.print(", ");
            }
        }
        System.out.print("]");
    }
}
```

6.Celebraty:

Language: Java 8

```java
import java.util.*;
public class Solution {
    public static int a(int[][] m, int n) {
        int c = 0;
        for (int i = 1; i < n; i++)
            if (m[c][i] == 1) c = i;
        for (int i = 0; i < n; i++)
            if (i != c && (m[c][i] == 1 || m[i][c] == 0)) return -1;
        return c;
    }
    public static void main(String[] a) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m[][] = new int[n][n];
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                m[i][j] = sc.nextInt();
        System.out.println((n = a(m, n)) == -1 ? "No Celebrity" : n);
    }
}
```

7. **Stack Permutation**

Main.java

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] input = new int[n], target = new int[n];
        for (int i = 0; i < n; i++) input[i] = sc.nextInt();
        for (int i = 0; i < n; i++) target[i] = sc.nextInt();
        Stack<Integer> st = new Stack<>();
        int j = 0;
        for (int i = 0; i < n; i++) {
            st.push(input[i]);
            while (!st.isEmpty() && st.peek() == target[j]) {
                st.pop();
                j++;
            }
        }
        System.out.println(st.isEmpty() ? "Yes" : "No");
    }
}
```

8.Tower of Hanoi

```java
import java.util.*;
public class Solution{
    public static void R(int n, char source, char target, char auxiliary) {
        if (n == 1) {
            System.out.println(source+" "+target);
            return;
        }
        R(n - 1, source, auxiliary, target);
        System.out.println(source+" "+target);
        R(n - 1, auxiliary, target, source);
    }

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        R(n, 'a', 'c', 'b');
    }
}

```

9. **Stock Span**

Language: Java 8

```java
import java.util.*;

class Solution {
    static void span(int n, int[] p, int[] s) {
        Stack<Integer> st = new Stack<>();
        for (int i = 0; i < n; i++) {
            while (!st.isEmpty() && p[st.peek()] <= p[i]) st.pop();
            s[i] = (st.isEmpty()) ? (i + 1) : (i - st.peek());
            st.push(i);
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] p = new int[n], s = new int[n];
        for (int i = 0; i < n; i++) p[i] = sc.nextInt();
        span(n, p, s);
        for (int val : s) System.out.print(val + " ");
    }
}

```

10.Minstack

Main.java

```java
import java.util.*;
public class Main {
    ArrayList<Integer> list = new ArrayList<>();
    int min = Integer.MAX_VALUE;

    void push(int x) {
        list.add(x);
        if (x < min) min = x;
    }

    int getMin() {
        return min;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Main ms = new Main();
        for (int n = sc.nextInt(); n-- > 0;) {
            int op = sc.nextInt();
            if (op == 1) ms.push(sc.nextInt());
            else if (op == 3) System.out.println(ms.getMin());
        }
    }
}

```

11.priority queue

Main.java

```java
import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] data = new int[n], pr = new int[n];

        for (int i = 0; i < n; i++) {
            data[i] = sc.nextInt();
            pr[i] = sc.nextInt();
        }

        Integer[] indices = new Integer[n];
        for (int i = 0; i < n; i++) indices[i] = i;

        Arrays.sort(indices, Comparator.comparingInt(i -> pr[i]));

        for (int i : indices) {
            System.out.println(data[i] + " " + pr[i]);
        }
    }
}

```

Time complexity:

Tower of hanoi (Recursive) - O(2^n), O(n)

Stack permutation (Stack) - O(n), O(n) ----- Brute force, recursive & back tracking - O(n!)

Stock span (Stack) - O(n), O(n) ---- Brute force - O(n^2),O(n); DP - O(n) but requires extra preprocessing

Sort without extra space (Selection sort+queue) - O(n^2),O(1) ---- bubble sort also same

Loop detection in ll (floyd's cycle) - O(n),O(1) ---- Hashing - O(n),O(n)

Celebrity problem (2-pointer approach) - O(n),O(1) ----- Stack - O(n),O(n); Brute force,matrix based - O(n^2),O(1);

Minimum stack - O(1),O(n)

Priority queue in dll - O(n),O(1)

Insert-O(n),deletion,peek-O(1) --- arrrays same, Heaps(optimal) - insert/delete-O(logn),peek-O(1)

Segregate even/odd nodes - O(n),O(1)

Merge sort dll - O(n log n),O(log n)

Bitonic - O(n),O(1)