Program:8a

```
import java.util.*;

class CheapestFlight {
    static class Edge {
        int dest, cost;
        Edge(int d, int c) { dest = d; cost = c; }
    }

    static int findCheapestCost(int n, int[][] flights, int src, int dest) {
        List<List<Edge>> graph = new ArrayList<>();
        for (int i = 0; i < n; i++) graph.add(new ArrayList<>());
        for (int[] f : flights)
            graph.get(f[0]).add(new Edge(f[1], f[2]));

        int[] dist = new int[n];
        Arrays.fill(dist, Integer.MAX_VALUE);
        dist[src] = 0;

        PriorityQueue<int[]> pq = new PriorityQueue<>(Comparator.comparingInt(a -> a[1]));
        pq.offer(new int[]{src, 0});

        while (!pq.isEmpty()) {
            int[] cur = pq.poll();
            int city = cur[0], cost = cur[1];
            if (city == dest) return cost;

            for (Edge e : graph.get(city)) {
```

```
                int newCost = cost + e.cost;
                if (newCost < dist[e.dest]) {
                    dist[e.dest] = newCost;
                    pq.offer(new int[]{e.dest, newCost});
                }
            }
        }
        return -1; // if path not found
    }

    public static void main(String[] args) {
        int n = 5;
        int[][] flights = {
            {0, 1, 100}, {1, 2, 100},
            {0, 2, 500}, {2, 3, 200},
            {3, 4, 100}
        };
        int src = 0, dest = 4;
        System.out.println("Cheapest cost: " + findCheapestCost(n, flights, src, dest));
    }
}
```

Output:

Cheapest cost: 500

Program:8b

```
import java.util.*;

class ConnectGroups {
    public static int connectTwoGroups(List<List<Integer>> cost) {
        int m = cost.size(), n = cost.get(0).size();
        int[] minCostB = new int[n];
        Arrays.fill(minCostB, Integer.MAX_VALUE);

        for (List<Integer> row : cost)
            for (int j = 0; j < n; j++)
                minCostB[j] = Math.min(minCostB[j], row.get(j));

        return dfs(cost, 0, 0, new HashMap<>(), minCostB);
    }

    private static int dfs(List<List<Integer>> cost, int i, int mask,
                           Map<String, Integer> memo, int[] minCostB) {
        int m = cost.size(), n = cost.get(0).size();
        if (i == m) {
            int extra = 0;
            for (int j = 0; j < n; j++)
                if ((mask & (1 << j)) == 0) extra += minCostB[j];
            return extra;
        }

        String key = i + "," + mask;
        if (memo.containsKey(key)) return memo.get(key);
```

```
        int ans = Integer.MAX_VALUE;
        for (int j = 0; j < n; j++) {
            ans = Math.min(ans, cost.get(i).get(j) + dfs(cost, i + 1, mask | (1 << j), memo,
minCostB));
        }
        memo.put(key, ans);
        return ans;
    }

    public static void main(String[] args) {
        List<List<Integer>> cost = Arrays.asList(
            Arrays.asList(15, 96),
            Arrays.asList(36, 2)
        );
        System.out.println("Minimum total cost: " + connectTwoGroups(cost));
    }
}
```

Output:

Minimum total cost: 17

Program:8c

```
import java.util.*;


class DecodeString {
    public static String decodeString(String s) {
        Stack<Integer> countStack = new Stack<>();
        Stack<StringBuilder> stringStack = new Stack<>();
        StringBuilder current = new StringBuilder();
        int k = 0;


        for (char ch : s.toCharArray()) {
            if (Character.isDigit(ch)) {
                k = k * 10 + ch - '0';
            } else if (ch == '[') {
                countStack.push(k);
                stringStack.push(current);
                current = new StringBuilder();
                k = 0;
            } else if (ch == ']') {
                StringBuilder decoded = stringStack.pop();
                int repeatTimes = countStack.pop();
                for (int i = 0; i < repeatTimes; i++)
                    decoded.append(current);
                current = decoded;
            } else {
                current.append(ch);
            }
        }
```

```
        return current.toString();
    }

    public static void main(String[] args) {
        String encoded = "3[a2[c]]";
        System.out.println("Decoded String: " + decodeString(encoded));
    }
}
```

Output:

Decoded String: accaccacc