# **DSA Problems on 31st May (Face Prep)**

# 1. Merge K sorted linked lists

CODE:

import java.io.*\**;

class ListNode {

    int value;

    ListNode next;

    ListNode(int val) {

        value = val;

        next = null;

    }

}

class mergeksortedlinkedlist {

    static ListNode head;

    static ListNode temp;

    static void printList(ListNode node) {

*while* (node != null) {

            System.out.print(node.value + " ");

            node = node.next;

        }

        System.out.println();

    }

    static ListNode mergeKLists(ListNode lists[], int last) {

*for* (int i = 1; i <= last; i++) {

*while* (true) {

                ListNode head0 = lists[0];

                ListNode headI = lists[i];

*if* (headI == null)

*break*;

*if* (head0.value >= headI.value) {

                    lists[i] = headI.next;

                    headI.next = head0;

                    lists[0] = headI;

                } *else* {

*while* (head0.next != null) {

*if* (head0.next.value >= headI.value) {

                            lists[i] = headI.next;

                            headI.next = head0.next;

                            head0.next = headI;

*break*;

                        }

                        head0 = head0.next;

*if* (head0.next == null) {

                            lists[i] = headI.next;

                            headI.next = null;

                            head0.next = headI;

                            head0.next.next = null;

*break*;

                        }

                    }

                }

            }

        }

*return* lists[0];

    }

    public static void main(String[] args) {

        int k = 3;

        int n = 4;

        ListNode[] lists = *new* ListNode[k];

        lists[0] = *new* ListNode(1);

        lists[0].next = *new* ListNode(3);

        lists[0].next.next = *new* ListNode(5);

        lists[0].next.next.next = *new* ListNode(7);

        lists[1] = *new* ListNode(2);

        lists[1].next = *new* ListNode(4);

        lists[1].next.next = *new* ListNode(6);

        lists[1].next.next.next = *new* ListNode(8);

        lists[2] = *new* ListNode(0);

        lists[2].next = *new* ListNode(9);

        lists[2].next.next = *new* ListNode(10);

        lists[2].next.next.next = *new* ListNode(11);

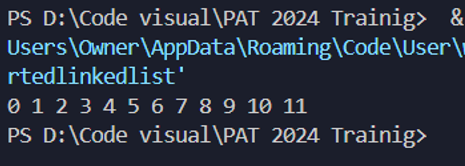
        head = mergeKLists(lists, k - 1);

        printList(head);

    }

}

OUTPUT:



# 2. Implement Stack in Java Using Array

CODE:

import java.io.*\**;

import java.util.*\**;

class GenericStack<T> {

    ArrayList<T> stackArray;

    int top = -1;

    int maxSize;

    GenericStack(int maxSize) {

*this*.maxSize = maxSize;

*this*.stackArray = *new* ArrayList<T>(maxSize);

    }

    void push(T element) {

*if* (top + 1 == maxSize) {

            System.out.println("Stack Overflow");

        } *else* {

            top = top + 1;

*if* (stackArray.size() > top) {

                stackArray.set(top, element);

            } *else* {

                stackArray.add(element);

            }

        }

    }

    T peek() {

*if* (top == -1) {

            System.out.println("Stack Underflow");

*return* null;

        } *else* {

*return* stackArray.get(top);

        }

    }

    void pop() {

*if* (top == -1) {

            System.out.println("Stack Underflow");

        } *else* {

            top--;

        }

    }

    boolean isEmpty() {

*return* top == -1;

    }

    @Override

    public String toString() {

        StringBuilder result = *new* StringBuilder();

*for* (int i = 0; i < top; i++) {

            result.append(stackArray.get(i)).append("->");

        }

        result.append(stackArray.get(top));

*return* result.toString();

    }

}

public class stackfromarray {

    public static void main(String[] args) {

        GenericStack<Integer> intStack = *new* GenericStack<>(3);

        intStack.push(10);

        intStack.push(20);

        intStack.push(30);

        System.out.println("intStack after pushing 10, 20 and 30 :\n" + intStack);

        intStack.pop();

        System.out.println("intStack after pop :\n" + intStack);

        GenericStack<String> stringStack = *new* GenericStack<>(3);

        stringStack.push("hello");

        stringStack.push("world");

        stringStack.push("java");

        System.out.println("\nstringStack after pushing 3 elements :\n" + stringStack);

        System.out.println("stringStack after pushing 4th element :");

        stringStack.push("okay done");

        GenericStack<Float> floatStack = *new* GenericStack<>(2);

        floatStack.push(100.0f);

        floatStack.push(200.0f);

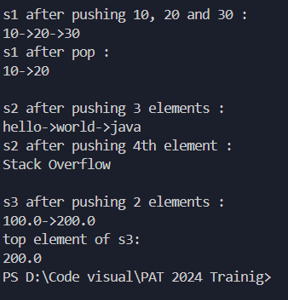
        System.out.println("\nfloatStack after pushing 2 elements :\n" + floatStack);

        System.out.println("top element of floatStack:\n" + floatStack.peek());

    }

}

OUTPUT:



# 3. Implement Stack using Queues

CODE:

import java.util.*\**;

class stackfromqueue {

    static class Stack {

        static Queue<Integer> queue1 = *new* LinkedList<Integer>();

        static Queue<Integer> queue2 = *new* LinkedList<Integer>();

        static void push(int element) {

            queue2.add(element);

*while* (!queue1.isEmpty()) {

                queue2.add(queue1.peek());

                queue1.remove();

            }

            Queue<Integer> temp = queue1;

            queue1 = queue2;

            queue2 = temp;

        }

        static void pop() {

*if* (!queue1.isEmpty()) {

                queue1.remove();

            }

        }

        static int top() {

*if* (queue1.isEmpty()) {

*return* -1;

            }

*return* queue1.peek();

        }

        static int size() {

*return* queue1.size();

        }

    }

    public static void main(String[] args) {

        Stack stack = *new* Stack();

        stack.push(1);

        stack.push(2);

        stack.push(3);

        System.out.println("Current size: " + stack.size());

        System.out.println(stack.top());

        stack.pop();

        System.out.println(stack.top());

        stack.pop();

        System.out.println(stack.top());

        System.out.println("Current size: " + stack.size());

    }

}

OUTPUT:

