```
import java.util.Comparator;
import java.util.PriorityQueue;;


public class runner_OfMKSL {

    public static void main(String[] args) {
        int k=3;
        Node[] arr = new Node[k];
        arr[0] = new Node(1);
        arr[0].link = new Node(2);
        arr[0].link.link = new Node(3);
        arr[0].link.link.link = new Node(4);

        arr[1] = new Node(5);
        arr[1].link=new Node(6);
        arr[1].link=new Node(7);

        arr[2] = new Node(8);
        arr[2].link = new Node(9);

        Node final_head = MKSL(arr, k);
        print_list(final_head);
    }


    static class Node {
        int item;
        Node link;
        public Node(int item) {
            this.item = item;
            link = null;
        }
    }


    public static Node add_item(int k) {
        Node new_node = new Node(k);
        new_node.link = null;
        return new_node;
    }


    public static void print_list(Node head) {
        if (!(head == null)) {
            System.out.print(head.item + " ");
            print_list(head.link);
        }
    }
```

```
    private static Node MKSL(Node[] arr, int k) {
        Node head = null, last = null;
        PriorityQueue<Node> min_heap = new PriorityQueue<>(
            new Comparator<Node>(){
                public int compare(Node a, Node b) {
                    return a.item - b.item;
                }
            }
        );

        for (int i=0; i<k; i++) if (arr[i] != null) min_heap.add(arr[i]);
        while (!min_heap.isEmpty()) {
            Node top = min_heap.peek();
            min_heap.remove();
            if (top.link != null) min_heap.add(top.link);
            if (head == null) head = last = top;
            else {
                last.link = top;
                last = top;
            }
        }
        return head;
    }

}
```