5a)

import java.util.Scanner;

class InsertionSortLinkedList {

static class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

public void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

return;

}

Node temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

}

public void printList(Node node) {

while (node != null) {

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

node = node.next;

}

System.out.println();

}

public Node insertionSort(Node head) {

if (head == null || head.next == null)

return head;

Node dummy = new Node(0);

Node curr = head;

while (curr != null) {

Node prev = dummy;

while (prev.next != null && prev.next.data < curr.data) {

prev = prev.next;

}

Node nextTemp = curr.next;

curr.next = prev.next;

prev.next = curr;

curr = nextTemp;

}

return dummy.next;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

InsertionSortLinkedList list = new InsertionSortLinkedList();

System.out.print("Enter number of elements: ");

int n = sc.nextInt();

System.out.println("Enter " + n + " elements:");

for (int i = 0; i < n; i++) {

int value = sc.nextInt();

list.insert(value);

}

System.out.print("Original List: ");

list.printList(list.head);

list.head = list.insertionSort(list.head);

System.out.print("Sorted List: ");

list.printList(list.head);

sc.close();

}

}

5b)

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void add(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

return;

}

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

public void remove(int key) {

if (head == null) {

System.out.println("List is empty. Nothing to remove.");

return;

}

if (head.data == key) {

head = head.next;

return;

}

Node current = head;

Node previous = null;

while (current != null && current.data != key) {

previous = current;

current = current.next;

}

if (current == null) {

System.out.println("Element " + key + " not found in the list.");

return;

}

previous.next = current.next;

}

public void display() {

Node current = head;

while (current != null) {

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

}

public class RemoveElementLinkedList {

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.add(1);

list.add(2);

list.add(3);

list.add(4);

list.add(5);

System.out.print("Original List: ");

list.display();

list.remove(3);

System.out.print("Updated List (after removing 3): ");

list.display();

list.remove(1);

System.out.print("Updated List (after removing 1): ");

list.display();

list.remove(6);

System.out.print("Updated List (after attempting to remove 6): ");

list.display();

}

}

5c)

import java.util.HashSet;

class Node {

    int data;

    Node next;

    Node(int data) {

        this.data = data;

        this.next = null;

    }

}

class SinglyLinkedList {

    Node head;

    public void add(int data) {

        Node newNode = new Node(data);

        if (head == null) {

            head = newNode;

            return;

        }

        Node current = head;

        while (current.next != null) {

            current = current.next;

        }

        current.next = newNode;

    }

    public void removeDuplicates() {

        if (head == null) return;

        HashSet<Integer> set = new HashSet<>();

        Node current = head;

        Node prev = null;

        while (current != null) {

            if (set.contains(current.data)) {

                prev.next = current.next;

            } else {

                set.add(current.data);

                prev = current;

            }

            current = current.next;

        }

    }

    public void display() {

        Node current = head;

        while (current != null) {

            System.out.print(current.data + " ");

            current = current.next;

        }

        System.out.println();

    }

}

public class RemoveDuplicatesLinkedList {

    public static void main(String[] args) {

        SinglyLinkedList list = new SinglyLinkedList();

        list.add(1);

        list.add(2);

        list.add(2);

        list.add(3);

        list.add(4);

        list.add(4);

        list.add(5);

        System.out.print("Original List: ");

        list.display();

        list.removeDuplicates();

        System.out.print("List After Removing Duplicates: ");

        list.display();

    }

}

6a)

import java.util.Scanner;

public class CountingSort {

public static void countingSort(int arr[], int n) {

int max = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] > max)

max = arr[i];

}

int count[] = new int[max + 1];

for (int i = 0; i <= max; i++)

count[i] = 0;

for (int i = 0; i < n; i++)

count[arr[i]]++;

for (int i = 1; i <= max; i++)

count[i] += count[i - 1];

int output[] = new int[n];

for (int i = n - 1; i >= 0; i--) {

output[count[arr[i]] - 1] = arr[i];

count[arr[i]]--;

}

for (int i = 0; i < n; i++)

arr[i] = output[i];

}

public static void printArray(int arr[], int n) {

for (int i = 0; i < n; i++)

System.out.print(arr[i] + " ");

System.out.println();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of elements: ");

int n = sc.nextInt();

int arr[] = new int[n];

System.out.println("Enter elements:");

for (int i = 0; i < n; i++) {

arr[i] = sc.nextInt();

}

System.out.println("Original array:");

printArray(arr, n);

countingSort(arr, n);

System.out.println("Sorted array:");

printArray(arr, n);

}

}

6b)

import java.util.Scanner;

public class RadixSort {

static int getMax(int arr[], int n) {

int max = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] > max)

max = arr[i];

}

return max;

}

static void countingSort(int arr[], int n, int exp) {

int output[] = new int[n];

int count[] = new int[10];

for (int i = 0; i < 10; i++)

count[i] = 0;

for (int i = 0; i < n; i++)

count[(arr[i] / exp) % 10]++;

for (int i = 1; i < 10; i++)

count[i] += count[i - 1];

for (int i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

for (int i = 0; i < n; i++)

arr[i] = output[i];

}

static void radixSort(int arr[], int n) {

int m = getMax(arr, n);

for (int exp = 1; m / exp > 0; exp \*= 10)

countingSort(arr, n, exp);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of elements: ");

int n = sc.nextInt();

int arr[] = new int[n];

System.out.println("Enter elements:");

for (int i = 0; i < n; i++) {

arr[i] = sc.nextInt();

}

System.out.println("Original array:");

printArray(arr, n);

radixSort(arr, n);

System.out.println("Sorted array:");

printArray(arr, n);

}

6c)

import java.util.Scanner;

public class HeapSort {

static void heapify(int arr[], int n, int i) {

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && arr[left] > arr[largest])

largest = left;

if (right < n && arr[right] > arr[largest])

largest = right;

if (largest != i) {

int temp = arr[i];

arr[i] = arr[largest];

arr[largest] = temp;

heapify(arr, n, largest);}

static void heapSort(int arr[], int n) {

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 printArray(int arr[], int n) {

for (int i = 0; i < n; i++)

System.out.print(arr[i] + " ");

System.out.println();

} 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("Original array:");

printArray(arr, n);

heapSort(arr, n);

System.out.println("Sorted array:");

printArray(arr, n); }

}