import java.util.Scanner;

public class Assign6 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of students in the second year: ");

        int numStudents = scanner.nextInt();

        System.out.println("Enter marks for each student:");

        int[] marks = new int[numStudents];

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

            marks[i] = scanner.nextInt();

        }

        int maxMarks = findMaxMarks(marks);

        int minMarks = findMinMarks(marks);

        System.out.println("Maximum Marks: " + maxMarks);

        System.out.println("Minimum Marks: " + minMarks);

    }

    public static int findMaxMarks(int[] marks) {

        int heapSize = marks.length;

        for (int i = (heapSize / 2) - 1; i >= 0; i--) {

            heapifyMax(marks, heapSize, i);

        }

        int maxMark = marks[0];

        return maxMark;

    }

    public static int findMinMarks(int[] marks) {

        int heapSize = marks.length;

        for (int i = (heapSize / 2) - 1; i >= 0; i--) {

            heapifyMin(marks, heapSize, i);

        }

        int minMark = marks[0];

        return minMark;

    }

    private static void heapifyMax(int[] arr, int heapSize, int parentIndex) {

        int largest = parentIndex;

        int leftChild = 2 \* parentIndex + 1;

        int rightChild = 2 \* parentIndex + 2;

        if (leftChild < heapSize && arr[leftChild] > arr[largest]) {

            largest = leftChild;

        }

        if (rightChild < heapSize && arr[rightChild] > arr[largest]) {

            largest = rightChild;

        }

        if (largest != parentIndex) {

            int temp = arr[parentIndex];

            arr[parentIndex] = arr[largest];

            arr[largest] = temp;

            heapifyMax(arr, heapSize, largest);

        }

    }

    private static void heapifyMin(int[] arr, int heapSize, int parentIndex) {

        int smallest = parentIndex;

        int leftChild = 2 \* parentIndex + 1;

        int rightChild = 2 \* parentIndex + 2;

        if (leftChild < heapSize && arr[leftChild] < arr[smallest]) {

            smallest = leftChild;

        }

        if (rightChild < heapSize && arr[rightChild] < arr[smallest]) {

            smallest = rightChild;

        }

        if (smallest != parentIndex) {

            int temp = arr[parentIndex];

            arr[parentIndex] = arr[smallest];

            arr[smallest] = temp;

            heapifyMin(arr, heapSize, smallest);

        }

    }

}