Worksheet 3

Non Prima

Deskripsi

Wondi adalah seorang mahasiswa semester tiga yang sedang mempelajari mata kuliah Struktur Data dan Algoritma. Dikarenakan liburannya yang kemarin terlalu lama, Wondi telah melupakan banyak hal mengenai materi-materi di semester-semester lalu. Suatu hari, kekasih Wondi bernama Alea yang merupakan mahasiswi semester tiga di Universitas lain meminta bantuan pada Wondi mengenai tugas pemrograman yang diterima oleh Alea. Awalnya, Wondi ingin menolak membantu Alea karena ia belum mempelajari lagi materi perkuliahannya semenjak liburan dimulai. Namun, karena Wondi memiliki gengsi setinggi langit dan ingin menjaga *image* di depan pacarnya, Wondi meminta Anda yang lebih jago dari Wondi dalam *programming* untuk membantunya menyelesaikan program tersebut.

Alea diminta membuat program yang dapat **mengurutkan angka berdasarkan jumlah dari faktor non-prima yang dimiliki angka tersebut**. Suatu bilangan dikatakan prima jika dan hanya jika bilangan tersebut mempunyai 2 faktor positif berbeda dan tidak mempunyai faktor positif selain 1 dan dirinya sendiri. **Perlu diperhatikan bahwa angka 1 bukanlah bilangan prima**. Angka-angka input sebanyak N akan diurutkan berdasarkan jumlah faktor non-prima terkecil ke jumlah faktor terbanyak. Bila ada lebih dari satu angka memiliki jumlah faktor non-prima yang sama, maka urutkan dari nilai angka yang paling kecil.

Pada worksheet ini, Anda **harus** mengimplementasikan algoritma sorting *bubble sort*. Pada *class* Number sudah ditambahkan interface Comparable. **Anda diwajibkan untuk mengimplementasikan algoritma *bubble sort* menggunakan method compareTo pada class Number**. Pada halaman berikutnya, Alea telah melampirkan template untuk memudahkan Anda dalam mengerjakan soal.

Format Masukan

Baris pertama berisi sebuah bilangan bulat N, yaitu banyak angka yang diberikan.
N baris berikutnya berisi sebuah bilangan bulat A_i, yaitu angka-angka yang akan diurutkan.

Format Keluaran

Keluarkan sebuah baris berisi N buah bilangan bulat yang dipisahkan oleh spasi yang diurutkan berdasarkan jumlah faktor yang dimiliki mulai dari jumlah paling sedikit ke jumlah paling banyak. Bila ada angka yang memiliki jumlah faktor yang sama, yang didahulukan adalah angka yang nilainya lebih kecil.

Contoh 1

Masukan

```
5
19
5
2
7
3
```

Keluaran

```
2 3 5 7 19
```

Penjelasan

Berikut adalah faktor-faktor dari setiap bilangan:
19 → **1**, 19 → 1 faktor non-prima
5 → **1**, 5 → 1 faktor non-prima
2 → **1**, 2 → 1 faktor non-prima
7 → **1**, 7 → 1 faktor non-prima
3 → **1**, 3 → 1 faktor non-prima

Contoh 2

Masukan

```
5
60
91
97
25
4
```

Keluaran

```
97 4 25 91 60
```

Penjelasan

Berikut adalah faktor-faktor non-prima dari setiap bilangan:
4 →**1**, 2, **4** → 2 faktor non-prima
25 → **1**, 5, **25** → 2 faktor non-prima
60 → **1**, 2, 3, **4**, 5, **6**, **10**, **12**, **15**, **20**, **30**, **60** → 9 faktor non-prima
91 → **1**, 7, 13, **91** → 2 faktor non-prima
97 → **1**, 97 → 1 faktor non-prima

Batasan Kasus Uji

Kasus uji dalam *grader* sudah terbagi dalam 5 tingkat kesulitan. Untuk semua kasus uji, berlaku:

- $1 \le N \le 5.000$
- $1 \le A_i \le 5.000$ untuk setiap $1 \le i \le N$

Sangat Mudah (3 kasus uji)

- $1 \le N \le 50$
- $1 \le A_i \le 100$ untuk setiap $1 \le i \le N$
- A_i dijamin merupakan bilangan prima
- *Hint*: Cukup implementasi *bubblesort* saja tanpa menghitung banyak faktor non-prima (*mengapa?*).

Mudah (6 kasus uji)

- $1 \le N \le 50$
- $1 \le A_i \le 100$ untuk setiap $1 \le i \le N$
- *Hint*: Solusi naif dapat dengan mudah lolos kasus uji ini.

Mudah-Menengah (3 kasus uji)

- $1 \le N \le 100$
- **$1 \le A_i \le 1.000$** untuk setiap $1 \le i \le N$
- *Hint*: Perhitungan banyak faktor non-prima harus '*lebih cepat*'.

Menengah (4 kasus uji)

- $1 \le N \le 1.000$
- $1 \le A_i \le 1.000$ untuk setiap $1 \le i \le N$
- *Hint*: Perlukah kita mengecek apakah suatu bilangan prima atau tidak berkali-kali?

Nilai 100 (2 kasus uji)

- Tidak ada batasan tambahan
- *Hint*: Apa lagi yang bisa dioptimasi seperti sebelumnya?

Catatan Khusus

Anda dapat membaca tentang dokumentasi *method* compareTo pada tautan berikut: https://www.tutorialspoint.com/java/number_compareto.htm atau Anda dapat mencari dokumentasi sendiri.

Template Kode

Perhatian: Anda boleh menambahkan/menghapus bagian kode apapun pada *template* (asalkan memenuhi aturan yang sudah diberikan di soal). Kami tidak menjamin bahwa solusi penuh (nilai 100) dapat dicapai hanya dengan mengisi bagian yang harus diisi. Link Template: https://pastebin.com/98FmGSc5

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class SortByNonPrimeFactors {
    public static void main(String args[]) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        int N = Integer.parseInt(reader.readLine());
        Number[] numbers = new Number[N];

        for (int i = 0; i < N; i++) {
            numbers[i] = new Number(Integer.parseInt(reader.readLine()));
        }

        bubbleSort(numbers);
        printHasilSorting(numbers);
    }

    static private void printHasilSorting(Number[] arr) {
        int n = arr.length;
        for (int i = 0; i < n; i++) {
            if (i != 0)
                System.out.print(' ');
            System.out.print(arr[i].num);
        }
        System.out.println();
    }

    static private void bubbleSort(Number[] arr) {
        int N = arr.length;
        Number temp;
        for(int i=0; i < N; i++) {
            for(int j=1; j < (N-i); j++){
                // @TODO: Implementasikan bubble sort Anda di sini
            }
        }
    }
}

class Number implements Comparable<Number> {
      // @TODO: lengkapi class dengan instance variable, constructor, dan method yang sesuai dan menurut Anda
diperlukan
    int num;

    public Number(int num) {
        this.num = num;
    }

    int findNonPrimeFactors() {
        // @TODO: lengkapi method ini untuk mencari faktor non prima dari sebuah bilangan
    }

    @Override
    public int compareTo(Number other) {
        // @TODO: lengkapi method untuk sorting sesuai dengan spesifikasi soal
    }
}
```