Tes Asisten Praktikum Alpro Lanjut

Tingkat kesulitan: Mudah
Estimasi waktu pengerjaan: 90 menit

Buatlah sebuah program yang akan meminta input “n” (1-300), dan menampilkan hasil penjumlahan dari setiap pembagi “n”. Nilai pembagi merupakan bilangan kecil dari n yang dapat membagi nilai n dengan hasil bagi berupa bilangan bulat.

Contoh jalannya program:

Masukkan nilai n: 20
Pembagi: 1, 2, 4, 5, 10
Hasil bagi: 22

Masukkan nilai n: 15
Pembagi: 1, 2, 3, 5
Hasil bagi: 11

Tingkat kesulitan: Mudah
Estimasi waktu pengerjaan: 120 menit

Buatlah sebuah program yang akan mentranslasikan bilangan basis n menjadi bilangan basis 10 (desimal). Program akan meminta dua input, yaitu bilangan basis 2 dan bilangan yang akan ditranslasikan. Pastikan melakukan validasi input sesuai basis yang diberikan.

Contoh jalannya program:

Masukkan basis bilangan: 2
Bilangan: 1001
Nilai: 9

Masukkan basis bilangan: 4
Bilangan: 32
Nilai: 14

Masukkan basis bilangan: 2
Bilangan: 10031
Nilai bilangan tidak sesuai dengan basis

Tingkat kesulitan: Sedang
Estimasi waktu pengerjaan: 120 menit

Diketahui sebuah program sebagai berikut:

```
public class NewClass {
    public static void main(String[] args) {
        int arrX = {23, 5, 17, 18, 3;
        for (int i = 0; i < arrX.length(); i++) {
            System.out.println("Array ke-"+i+": "+arrX[j]));
        }
    }
}
```

Carilah kesalahan dari program tersebut. Tuliskan semua kesalahan yang ada pada program di atas. Jangan tuliskan program yang sudah diperbaiki.

Tingkat kesulitan: Sedang
Estimasi waktu pengerjaan: 120 menit

Diketahui sebuah program sebagai berikut:

```
public class NewClass {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("Masukkan nama: ");
        String nama = s.nextLine();
        System.out.print("Masukkan prodi: ");
        String prodi = s.nextLine();
        System.out.print("Masukkan angkatan: ");
        int angkatan = s.nextInt();
        System.out.print("Masukkan peminatan: ");
        String minat = s.nextLine();

        System.out.println("\n\nNama: "+nama);
        System.out.println("Prodi: "+prodi);
        System.out.println("Angkatan: "+angkatan);
        System.out.println("Peminatan: "+minat);
    }
}
```

Hasil dari program di atas adalah sebagai berikut:

```
run:
Masukkan nama: Melody Nurramdhani
Masukkan prodi: D3 Manajemen Informatika
Masukkan angkatan: 2013
Masukkan peminatan:

Nama: Melody Nurramdhani
Prodi: D3 Manajemen Informatika
Angkatan: 2013
Peminatan:
BUILD SUCCESSFUL (total time: 23 seconds)
```

Terlihat masukan “peminatan” akan dilewatkan saat program dijalankan. Program tidak akan menunggu *user* untuk memasukkan “peminatan” setelah memasukkan nilai “angkatan”. Bagaimana caranya agar *user* tetap dapat memasukkan nilai dari “peminatan”, sehingga semua data dapat lengkap ditampilkan?

Modifikasilah program di atas, sehingga setiap “nama”, “prodi”, “angkatan”, dan “peminatan” dapat dimasukkan nilainya oleh user.

Tingkat kesulitan: Sulit
Estimasi waktu pengerjaan: 180 menit

Algoritma pencarian merupakan algoritma yang dimaksudkan untuk mencari data pada sebuah *list* data. Masukan dari algoritma biasanya adalah sebuah array, dan keluarannya berupa indeks tempat bilangan itu berada pada array. Setelah itu, dilakukan pencarian data dengan membagi dua keseluruhan data.

Contoh implementasi program yang menyatakan algoritma ini tertulis sebagai berikut (diambil dari http://java2novice.com/java-search-algorithms/linear-search/):

```
public class MyLinearSearch {

    public static int linerSearch(int[] arr, int key) {
        int size = arr.length;
        for (int i = 0; i < size; i++) {
            if (arr[i] == key) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String a[]) {
        int[] arr1 = {23, 45, 21, 55, 234, 1, 34, 90};
        int searchKey = 34;
        System.out.println("Key " + searchKey + " found at index: " +
linerSearch(arr1, searchKey));
        int[] arr2 = {123, 445, 421, 595, 2134, 41, 304, 190};
        searchKey = 421;
        System.out.println("Key " + searchKey + " found at index: " +
linerSearch(arr2, searchKey));
    }
}
```

Dengan kata-kata sendiri, jelaskanlah cara kerja dari program di atas, dan hubungkan dengan penjelasan dari algoritma pencarian sekuensial.

Tingkat kesulitan: Sulit
Estimasi waktu pengerjaan: 240 menit

Algoritma pencarian merupakan algoritma yang dimaksudkan untuk mencari data pada sebuah *list* data. Masukan dari algoritma biasanya adalah sebuah array, dan keluarannya berupa indeks tempat bilangan itu berada pada array.

Salah satu algoritma pencarian adalah algoritma pencarian biner (*binary search*). Syarat dari pencarian sekuensial adalah keterurutan data yang akan dicari pada array masukan. Setelah itu, dilakukan pencarian data dengan membagi dua keseluruhan data.

Contoh implementasi program yang menyatakan algoritma ini tertulis sebagai berikut (diambil dari http://java2novice.com/java-search-algorithms/binary-search/):

```
public class MyBinarySearch {

    public int binarySearch(int[] inputArr, int key) {
        int start = 0;
        int end = inputArr.length - 1;
        while (start <= end) {
            int mid = (start + end) / 2;
            if (key == inputArr[mid]) {
                return mid;
            }
            if (key < inputArr[mid]) {
                end = mid - 1;
            } else {
                start = mid + 1;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        MyBinarySearch mbs = new MyBinarySearch();
        int[] arr = {2, 4, 6, 8, 10, 12, 14, 16};
        System.out.println("Key 14's position: " + mbs.binarySearch(arr, 14));
        int[] arr1 = {6, 34, 78, 123, 432, 900};
        System.out.println("Key 432's position: " + mbs.binarySearch(arr1, 432));
    }
}
```

Dengan kata-kata sendiri, jelaskanlah cara kerja dari program di atas, dan hubungkan dengan penjelasan dari algoritma pencarian biner.

Tingkat kesulitan: Sulit
Estimasi waktu pengerjaan: 180 menit

Algoritma pengurutan merupakan algoritma yang bertujuan untuk mengurutkan data dari sebuah *list* yang tidak terurut. Biasanya, masukan dari algoritma ini adalah sebuah array tidak terurut, dan akan mengembalikan nilai array yang sudah terurut. Cara kerja dari algoritma pengurutan biasanya menggunakan perulangan/*looping*.

Salah satu contoh algoritma pengurutan adalah Selection Sort. Cara kerja dari Selection Sort adalah melakukan pemilihan bilangan terkecil dari array yang diketahui. Bilangan terkecil itu akan dipertukarkan letaknya dengan bilangan pada indeks pertama array. Daftar bilangan yang akan diproses akan berkurang seiring dengan perulangan yang dilakukan. Dengan kata lain, indeks "pertama" dari array akan berubah seiring bertambahnya jumlah perulangan yang dilakukan. Variasi dari Selection Sort, juga bisa diimplementasikan dengan menggunakan pemilihan bilangan terbesar.

Contoh implementasi dari Selection Sort diperluhatkan *listing code* berikut (diambil dari http://www.java2novice.com/java-sorting-algorithms/selection-sort/):

```
public class MySelectionSort {

    public static int[] doSelectionSort(int[] arr) {
        for (int i = 0; i < arr.length - 1; i++) {
            int index = i;
            for (int j = i + 1; j < arr.length; j++) {
                if (arr[j] < arr[index]) {
                    index = j;
                }
            }
            int smallerNumber = arr[index];
            arr[index] = arr[i];
            arr[i] = smallerNumber;
        }
        return arr;
    }

    public static void main(String a[]) {
        int[] arr1 = {10, 34, 2, 56, 7, 67, 88, 42};
        int[] arr2 = doSelectionSort(arr1);
        for (int i : arr2) {
            System.out.print(i);
            System.out.print(", ");
        }
    }
}
```

Dengan kata-kata sendiri, jelaskanlah cara kerja dari program di atas, dan hubungkan dengan penjelasan dari algoritma Selection Sort.