Nama		Kelas	
NPM		Nomor Meja	

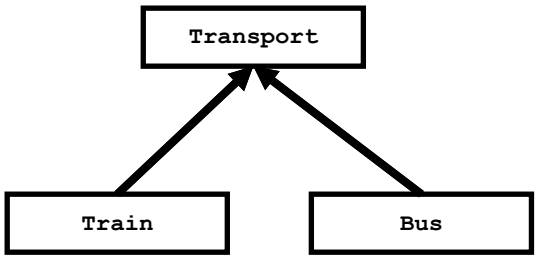
UJIAN TENGAH SEMESTER
CSF1600400 – STRUKTUR DATA DAN ALGORITMA - 2013
FAKULTAS ILMU KOMPUTER – UNIVERSITAS INDONESIA
Sabtu, 6 April 2013, Jam: 09.00 – 11.30
Sifat ujian: Open Notes

Perhatikan:

- *Ujian ini terdiri dari **5** bagian, jumlah soal ada **11 soal** dalam **18 halaman** dicetak bolak-balik . Periksa **jumlah soal** dan **halaman** sebelum menulis nama. Tuliskan nama, NPM, kelas dan nama dosen anda pada tiap lembar.*
- *Kerjakan soal yang anda anggap mudah terlebih dahulu.*
- *Nilai total: 110 (10 point bonus)*
- *Baca soal dengan **teliti** dan tulis jawaban Anda dengan tulisan yang **jelas** pada **tempat yang tersedia** pada lembaran yang sama dengan lembaran soal tersebut.*

BAGIAN I

1. (10 point) Di dalam sebuah program, terdapat class `Transport`, `Train`, **`Bus`**, dengan susunan inheritance seperti pada gambar berikut.



Kemudian, terdapat sebuah **`Main`** class yang memiliki method-method misterius berikut ini.

```
static void printMistery(List<Transport> arr){
     for(Transport t : arr)
        System.out.println(t);
}

static <T,R extends Transport> boolean findMistery(List<T> arr, R x) {
     for (T p : arr)
        if(x.equals(p)) return true;
     return false;
}
```

Nama		Kelas	
NPM		Nomor Meja	

a) Perhatikan pernyataan berikut:

"Ketika dilakukan pemanggilan method **`printMistery`** dengan parameter sebuah **`List<Train>`**, maka akan *compile-safe*, namun terjadi *runtime error* ketika dijalankan."

Apakah pernyataan tersebut benar? Berikan alasan.

b) Apa yang dilakukan oleh method **`findMistery`**?

c) Apa yang terjadi jika dilakukan pemanggilan method **`findMistery`** dengan parameter pertamanya adalah **`List<String>`**?

d) Pada saat melakukan pemanggilan method **`findMistery(list, item)`**, item boleh merupakan tipe apa saja?

2. (5 point) Berapakah kompleksitas dari deskripsi algoritma berikut ini?

Soal	Deskripsi Algoritma	Kompleksitas Algoritma
a)	Sebuah algoritma menerima input berukuran N. Algoritma ini membagi input tersebut menjadi dua bagian masing-masing 0.5 N dan memecahkannya secara rekursif. Operasi divide tersebut membutuhkan waktu O(1). Operasi conquer algoritma ini juga membutuhkan waktu O(1).	
b)	Sebuah algoritma menerima input berukuran N. Algoritma ini melakukan tiga buah komputasi secara berurutan: 1) Komputasi pertama membutuhkan waktu O(N) 2) Memanggil rekursif dengan input 0.5 N pertama 3) Memanggil rekursif dengan input 0.5 N sisanya. Proses divide pada komputasi 2) dan 3) membutuhkan waktu O(1). Kemudian dilakukan penggabungan hasil dari ketiga komputasi dengan waktu O(1).	

Nama		Kelas	
NPM		Nomor Meja	

3. (5 point) Tentukan perkiraan running time dari beberapa kasus input program berikut ini.

a)	Sebuah algoritma yang kompleksitasnya $O(N^2)$, membutuhkan waktu 5 detik untuk N = 100. Berapa waktu yang dibutuhkan untuk input N = 10000?
b)	Sebuah algoritma yang kompleksitasnya O(log N), membutuhkan waktu 10 detik untuk N = 32. Berapa waktu yang dibutuhkan untuk input N = 128?

Nama		**Kelas**	
NPM		**Nomor Meja**	

HALAMAN INI TIDAK BERISI SOAL

Nama		Kelas	
NPM		Nomor Meja	

BAGIAN II

4. (15 point) Tuliskan keluaran dari potongan program-program berikut ini.

```
public static void main (String args[])
{
    String[] arrString = { "saya", "pasti", "bisa", "lulus", "sda" };

    Stack<String> stack = new Stack<String> ();

    for (int i = 0; i < arrString.length; i++) {
        stack.push (arrString[i]);
    }
    for (int i = 0; i < arrString.length; i++) {
        System.out.print (stack.pop () + " ");
    }
    System.out.println ("");

    Queue<String> queue1 = new LinkedList<String> ();

    for (int i = 0; i < arrString.length; i++) {
        queue1.offer (arrString[i]);
    }
    for (int i = 0; i < arrString.length; i++) {
        System.out.print (queue1.poll () + " ");
    }
    System.out.println ("");

    Queue<String> queue2 = new PriorityQueue<String> ();

    for (int i = 0; i < arrString.length; i++) {
        queue2.add (arrString[i]);
    }
    for (int i = 0; i < arrString.length; i++) {
        System.out.print (queue2.poll () + " ");
    }
    System.out.println ("");
}
```

Jawaban:

Nama		Kelas	
NPM		Nomor Meja	

```
public static void main (String args[])
{
    String[] str = { "f", "a", "s", "i", "l", "k", "o", "m" };

    Queue<Comparable> queue = new PriorityQueue<Comparable> ();
    Stack<Comparable> stack = new Stack<Comparable> ();

    for (int i = 0; i < str.length; i++) {
        queue.offer (str[i]);
    }

    for (int i = 0; i < str.length; i++) {
        stack.push (queue.poll ());
    }

    while (!stack.isEmpty ()) {
        System.out.print (stack.pop () + " ");
    }
    System.out.println ("");
}
```

Jawaban:

Nama		Kelas	
NPM		Nomor Meja	

5. (10 point) Anda diminta untuk membuat suatu method yang menerima dua buah stack yang terurut A dan B (*min on top*) dan membuat sebuah stack gabungan yang sudah terurut (*min on top*). Anda hanya diperbolehkan untuk menggunakan operasi stack yaitu **pop(), push(), size()**, dan **top()**. Anda tidak diperkenankan menggunakan struktur data lain seperti array dan hanya diperbolehkan menggunakan stack. Elemen yang disimpan pada stack dapat dibandingkan satu sama lain dengan menggunakan fungsi **compareTo().**

Lengkapilah *method* berikut supaya berjalan dengan benar!

```
public <E extends Comparable<? super E>> Stack<E>
    mergeSortedStacks (Stack<E> A, Stack<E> B)
{









}
```

Nama		**Kelas**	
NPM		**Nomor Meja**	

HALAMAN INI TIDAK BERISI SOAL

Nama		Kelas	
NPM		Nomor Meja	

BAGIAN III

6. (10 point) Perhatikan program dibawah ini:

```
public class Soal6
{
    public static void main (String args[])
    {
        System.out.println (hitung (81));
    }

    public static int hitung (int x)
    {
        if (x <= 1) {
            return x;
        }
        return hitung (x, x / 2);
    }

    private static int hitung (int x, int y)
    {
        if (x <= 1) {
            return x;
        }
        if (y > x / y) {
            int z = ((x / y) + y) / 2;
            return hitung (x, z);
        } else {
            return y;
        }
    }
}
```

Apa output dari program diatas?

Disebut apa metode pendekatan pemecahan masalah pada method `hitung`?

Nama		**Kelas**	
NPM		**Nomor Meja**	

Ubahlah method hitung dari pendekatan rekursif ke iteratif

Nama		Kelas	
NPM		Nomor Meja	

BAGIAN IV

7. (10 point) Perhatikan program dibawah ini:

```
public static void misteri (int n) {
    for (int ii = 2; ii <= n; ii++) {
        int c = 0;
        for (int jj = 1; jj <= ii; jj++) {
            if (ii % jj == 0) {
                c++;
            }
        }
        if (c == 2) {
            System.out.print (ii + " ");
        }
    }
    System.out.println ();
}

public static void main (String[] args) {
    misteri (20);
}
```

Apa output dari program diatas?

Jelaskan dengan singkat (max 30 kata) apa yang dilakukan oleh method misteri (20)?

Jelaskan kompleksitas dari method misteri (n)?

Nama		Kelas	
NPM		Nomor Meja	

8. (15 point) Perhatikan program dibawah ini:

```
public class ListNode<E> {
    E data;
    ListNode<E> next;

    public ListNode (E data, ListNode<E> next) {
        this.data = data;
        this.next = next;
    }

    public ListNode (E data) {
        this (data, null);
    }
}

public class MidTestLinkedList {
    public static <E> void print (ListNode<E> h) {
        ListNode<E> c = h;
        while (c != null) {
            System.out.print (c.data + " ");
            c = c.next;
        }
    }

    public static <E> void misteri (ListNode<E> c) {
        if (c == null) return;
        misteri (c.next);
        System.out.print (c.data + " ");
    }

    public static <E> void misteri2 (ListNode<E> c) {
        ListNode<E> t = c.next;
        c.next = t.next;
        t.next = c.next.next;
        c.next.next = t;
    }

    public static void main (String[] args) {
        ListNode<String> h = new ListNode<String> ("X");
        h = new ListNode<String> ("Y", h);
        // baris 1
        print (h);
        System.out.println ();
        // baris 2
        misteri (h);
        System.out.println ();

        ListNode<String> j = new ListNode<String> ("A", new ListNode<String> (
                "B", new ListNode<String> ("C", new ListNode<String> ("D"))));
        // baris 3
        print (j);
        System.out.println ();
        // baris 4
        misteri2 (j);
        print (j);
        System.out.println ();
    }
}
```

Nama		Kelas	
NPM		Nomor Meja	

Program di atas mencetak 4 baris output. Tuliskan ke-empat baris output tersebut di bawah ini (11 point).

	Jawaban
1	
2	
3	
4	

(2 point) Jelaskan dengan singkat (max 30 kata) apa yang dilakukan oleh method `misteri()`?

(2 point) Jelaskan dengan singkat (max 30 kata) apa yang dilakukan oleh method `misteri2()`?

Nama		Kelas	
NPM		Nomor Meja	

9. (10 point) Lengkapi implementasi Stack dengan menggunakan Linked list. Implementasi ini tidak menggunakan header node. Gunakan class ListNode<E> yang diberikan pada soal sebelum ini (nomor 8).

```
public class StackLinkedList<E>
{
    // points to the top of stack
    ListNode<E> tos;

    // create new empty stack
    public StackLinkedList ()
    {
```

```
    }

    // returns true if the stack is empty
    public boolean isEmpty ()
    {
```

```
    }

    // clear the stack contents. make the stack empty.
    public void makeEmpty ()
    {
```

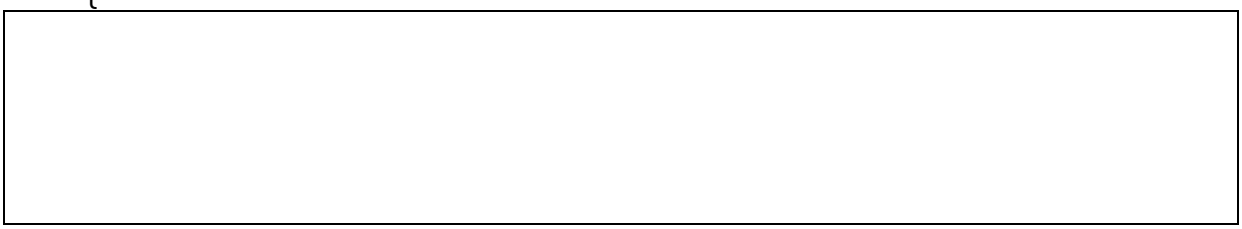
```
    }

    // access the top of this stack without removing
    // throw IllegalArgumentException if this stack is empty.
    public E top ()
    {
```

```
    }
```

Nama		Kelas	
NPM		Nomor Meja	

```
    // remove the last entry added to this stack
    // throw IllegalArgumentException if this stack is empty.
    public void pop ()
    {
```



```
    }

    // add new entry to this stack
    public void push (E x)
    {
```

```
    }
}
```

Nama		**Kelas**	
NPM		**Nomor Meja**	

HALAMAN INI TIDAK BERISI SOAL

Nama		Kelas	
NPM		Nomor Meja	

BAGIAN V

10. (10 point) Perhatikan rumus menghitung jumlah kombinasi r obyek dari n obyek dibawah ini:

$$C(n,r) = \frac{n!}{r!(n-r)!} = \frac{n}{r} \times \frac{n-1}{r-1} \times \cdots \times \frac{n-r+1}{1}$$

Rumus perhitungan tersebut tersebut bisa dituliskan secara rekursif di bawah ini:

$$C(n,0) = 1$$
$$C(n,n) = 1$$
$$C(n,r) = C(n-1,r-1) + C(n-1,r), \quad jika\ 0 < r < n$$

Lengkapi method **REKURSIF** kombinasi dibawah ini agar mendapatkan hasil sesuai rumus diatas:

```
public class Soal10
{
    public static void main (String args[]) {
        System.out.println (kombinasi (6, 4)); //cetak 15
    }

    //Lengkapi method rekursif dibawah ini
    static double kombinasi (int n, int r) {









    }
}
```

Apakah implementasi rekursif di atas sudah efisien? Jika belum efisien, jelaskan cara untuk menghitung jumlah kombinasi yang lebih efisien (ide-nya saja, tidak perlu menuliskan program-nya).

Nama		Kelas	
NPM		Nomor Meja	

11. (10 point) Perhatikan program dibawah ini:

```
public class Soal11
{
    public static void main (String args[]) {
        int data[] = { 1, 2, 5 };
        System.out.println (doSomething (data, data.length, 3));
        System.out.println (doSomething (data, data.length, 4));
    }

    static boolean doSomething (int data[], int n, int x) {
        boolean table[][] = new boolean[x + 1][n + 1];
        for (int i = 0; i <= n; i++) {
            table[0][i] = true;
        }
        for (int i = 1; i <= x; i++) {
            table[i][0] = false;
        }
        for (int i = 1; i <= x; i++) {
            for (int j = 1; j <= n; j++) {
                table[i][j] = table[i][j - 1];
                if (i >= data[j - 1]) {
                    table[i][j] = table[i][j] || table[i - data[j - 1]][j - 1];
                }
            }
        }
        return table[x][n];
    }
}
```

Apa output dari program diatas?

Jelaskan dengan singkat (max 30 kata) apa yang dilakukan oleh method doSomething?

Jelaskan kompleksitas dari method doSomething?

Pendekatan apa yang digunakan pada method doSomething?