LAPORAN JOBSHEET 5

KONSEP DASAR PEMROGRAMAN

Mata Kuliah : Algoritma dan Struktur Data

Dosen : **Mungki Astiningrum, S.T., M.Kom.**

Alfreda Dhaifullah Mahezwara
244107020219
Kelas : 1A
Absen : 04

PROGRAM STUDI TEKNIK INFORMATIKA JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG TAHUN 2025

Percobaan 1

```
package Pertemuan_5;


import java.util.Scanner;
public class bruteForceDivideConquer {


    public static void main(String[] args) {
        Scanner str = new Scanner (System.in);
        System.out.print("Masukan nilai: ");
        int nilai = str.nextInt();


        Faktorial fk = new Faktorial();
        System.out.println("Nilai faktorial " + nilai + "menggunakan BF: " +
fk.faktorialBF(nilai));
        System.out.println("Nilai faktorial " + nilai+ "menggunakan DC: " +
fk.faktorialDC(nilai));


    }
}


package Pertemuan_5;


public class Faktorial {
    int faktor;


    int faktorialBF (int n) {
        int fakto =1;
        for (int i = 1; i <= n; i++) {
            fakto = fakto * i;
        }


        return fakto;
    }


    int faktorialDC (int n) {
        if (n ==1) {
            return 1;
        } else {
```

```
            int fakto = n *faktorialDC(n-1);
            return fakto;
        }
    }
}
```

Pertanyaan

1. Pada base line algoritma Divide Conquer untuk melakukan pencarian nilai factorial, jelaskan perbedaan bagian kode pada penggunaan if else !
   - Bagian if (n == 1) merupakan kondisi penghentian base case
   - Bagian else merupakan bagian algoritma DIvid and Conquer.
2. Apakah memungkinkan perulangan pada method faktorialBF() diubahselainmenggunakan for? Buktikan!
   - Ya,

```
int faktorialBF (int n) {
        int fakto =1;
        int i = 1;
        do {
            fakto = fakto * i;
            i++;
        } while (i <= n);

        return fakto;
    }
```

3. Jelaskan perbedaan antara fakto *=i; dan int fakto n * faktorialDC (n - 1); !
   - Fakto*=I menggunakan perulangan untuk menghitung factorial
   - Int fakto = n * faktorialDC(n-1); menggunakan rekursi untuk menghitung factorial
4. Buat kesimpulan tentang perbedaan cara kerja method faktorialBF() dan factorialDC()!
   - Pada method factorialBF() menggunakan loop untuk menghitung nilai factorial, sementara method faltorialDC menggunakan rekursi untuk menghitung nilai faktorial

Percobaan 2

```
package Pertemuan_5;


import java.util.Scanner;
public class mainPangkat {
    public static void main (String []args) {
        Scanner str = new Scanner (System.in);
        System.out.println("Masukan jumlah elemen: ");
        int element = str.nextInt();


        Pangkat[] png = new Pangkat[element];
        for (int i = 0; i < element; i++) {
```

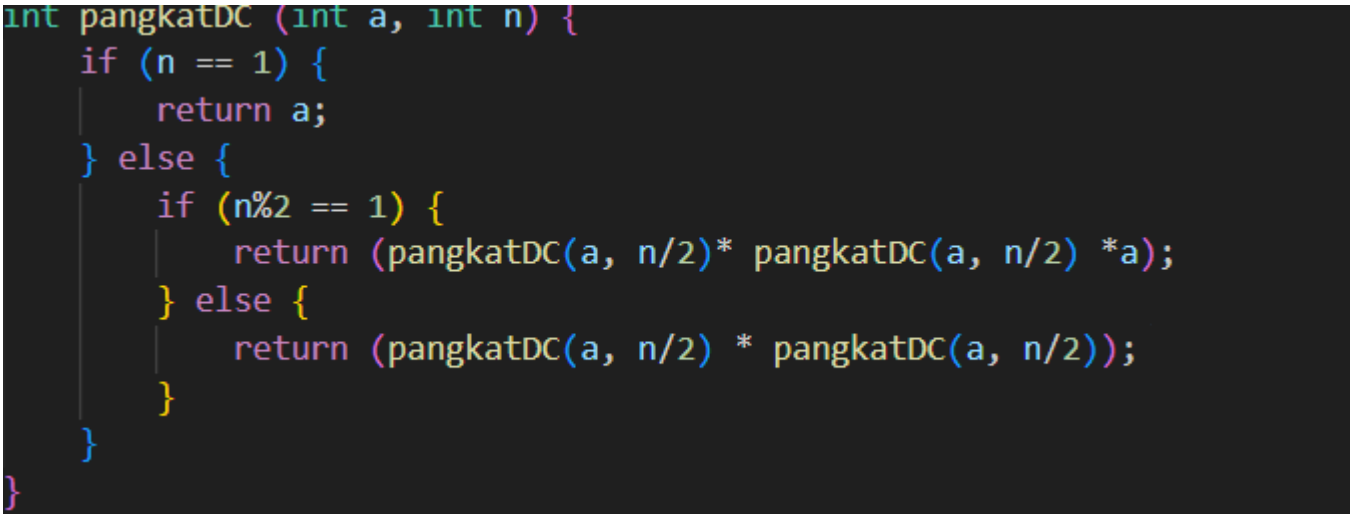
```
            System.out.print("Masukan nilai basis elemen ke-" + (i+1)+": ");
            int basis = str.nextInt();
            System.out.print("Masukan nilai pangkatelemen ke-" +(i+1)+": ");
            int pangkat = str.nextInt();
            png[i] = new Pangkat(basis, pangkat);
        }

        System.out.println("HASIL PANGKAT BRUTEFORCE:");
        for (Pangkat p: png) {
            System.out.println(p.nilai +"^"+ p.pangkat+": " +
p.pangkatBF(p.nilai,p.pangkat));
        }

        System.out.println("HASIL PANGKAT DIVIDE AND CONQUER: ");
        for (Pangkat p : png) {
            System.out.println(p.nilai+"^"+p.pangkat+":
"+p.pangkatDC(p.nilai,p.pangkat));
        }
    }
}
```

```
package Pertemuan_5;

public class Pangkat {
    int nilai, pangkat;

    Pangkat(int n, int p) {
        nilai = n;
        pangkat = p;
    }

    int pangkatBF (int a, int n) {
        int hasil = 1;
        for (int i = 0; i < n ;i++) {
            hasil = hasil * a;
        }
        return hasil;
    }
```

```
    int pangkatDC (int a, int n) {
        if (n == 1) {
            return a;
        } else {
            if (n%2 == 1) {
                return (pangkatDC(a, n/2)* pangkatDC(a, n/2) *a);
            } else {
                return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
            }
        }
    }
}
```

Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
   - Pada pangkatBF menggunakan loop untuk menghitung pangkat
   - Pada pangkataDC menggunakan rekursi untuk menghitung pangkat
2. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!



```
int pangkatDC (int a, int n) {
    if (n == 1) {
        return a;
    } else {
        if (n%2 == 1) {
            return (pangkatDC(a, n/2)* pangkatDC(a, n/2) *a);
        } else {
            return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
        }
    }
}
```

3. Pada method pangkatBF()terdapat parameter untuk melewatkan nilai yang akan dipangkatkan dan pangkat berapa, padahal di sisi lain di class Pangkat telah adaatribut nilai dan pangkat, apakah menurut Anda method tersebut tetap relevan untuk memiliki parameter? Apakah bisa jika method tersebut dibuat dengan tanpa parameter? Jikabisa, seperti apa method pangkatBF() yang tanpa parameter?
   - Tidak diperlukan,

```
int pangkatBF () {
        int hasil = 1;
        for (int i = 0; i < this.pangkat ;i++) {
            hasil = hasil * this.nilai;
        }
        return hasil;
    }
```

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
    - pangkatBF() menggunakan perulangan untuk menghitung pangkat
    - metode inin menggunakan fungsi rekursif dengan pendekatan Divide and Conquer

Percobaan 3

```
package Pertemuan_5;


import java.util.Scanner;
public class mainSum {
    public static void main(String[] args) {
        Scanner str = new Scanner (System.in);
        System.out.print("Masukan jumlah elemen");
        int elemen = str.nextInt();


        classSum sm = new classSum(elemen);
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukan keuntungan ke-"+ (i+1)+ ": ");
            sm.keuntungan[i] = str.nextDouble();
        }


        System.out.println("Total keuntungan Brute Force: " + sm.totalBF());
        System.out.println("Total keuntungan menggunakan DIvide and conquer: " +
sm.totalDC(sm.keuntungan, 0, elemen-1));
    }
}
```

```
package Pertemuan_5;


public class classSum {
    double keuntungan[];


    // konstruktor berparamater
    classSum (int el) {
        keuntungan = new double[el];
    }
```

```
    // menthod menghitung total nilai array dengan iterative
    double totalBF () {
        double total = 0;
        for (int i = 0; i < keuntungan.length; i++) {
            total = total+keuntungan[i];
        }
        return total;
    }

    double totalDC (double arr[], int l, int r) {
        if (l == r) {
            return arr[l];
        }

        int mid = (l+r)/2;
        double lsum = totalDC(arr, l, mid);
        double rsum = totalDC(arr, mid+1, r);
        return lsum + rsum;
    }
}
```

Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
   - Variable mid digunakan untuk membagi array menjadi dua bagian dalam proses Divide and Conquer
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?
   - Statement ini digunakan untuk menghitung keuntungan pada dua bagian array.
3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?
   - Karena telah dilakukan proses membagi array menjadi dua bagian, maka untuk mendapatkan total keseluruhan harus menjumlahkan hasil dari dua bagian tersebut
4. Apakah base case dari totalDC()?
   - If (l == r) { return arr[l] };
5. Tarik Kesimpulan tentang cara kerja totalDC()
   - Array dibagi menjadi dua bagian, masing masing bagian menghitung rekursif, hasil dari dua bagian dijumlahkan kembali untuk mendapat total keseluruhan

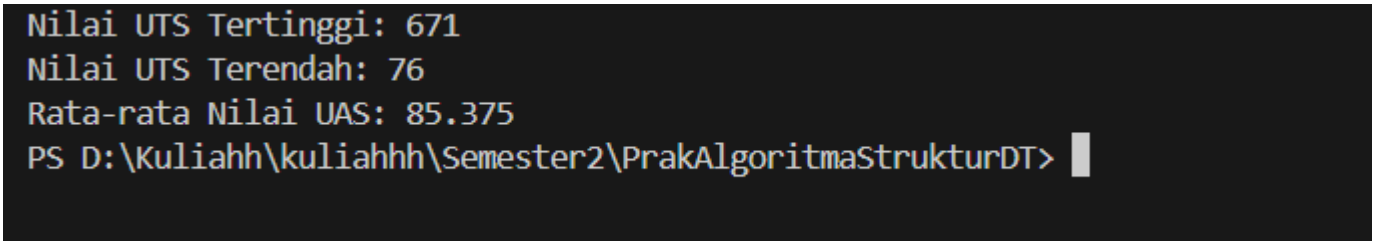
Pertanyaan

```
package Pertemuan_5;
```

```
public class nilaiMahasiswa {
    // Data nilai UTS dan UAS
    static int[] nilaiUTS = {78, 85, 90, 76, 92, 88, 80, 82};
    static int[] nilaiUAS = {82, 88, 87, 79, 95, 85, 83, 84};

    // mencari nilai UTS tertinggi dengan Divide and Conquer
    public static int UTStertinggi(int[] arr, int a, int l) {
        if (a == l) {
            return arr[a];
        }
        int mid = (a + l) / 2;
        int leftMax = UTStertinggi(arr, a, mid);
        int rightMax = UTStertinggi(arr, mid + 1, l);
        return leftMax + rightMax;
    }

    //mencari nilia UTS terendah dengan menggunakan Divide
    public static int UTSterendah(int[] arr, int b, int c) {
        if (b == c) {
            return arr[b];
        }
        int mid = (b + c) / 2;
        int leftMin = UTSterendah(arr, b, mid);
        int rightMin = UTSterendah(arr, mid + 1, c);
        return Math.min(leftMin, rightMin);
    }

    // Brute Force untuk mencari rata-rata nilai UAS
    public static double RataUAS(int[] arr) {
        int total = 0;
        for (int i = 0; i < arr.length; i++ ) {
            total += arr[i];
        }
        return (double) total / arr.length;
    }

    public static void main(String[] args) {
```

```
        int maxUTS = UTStertinggi(nilaiUTS, 0, nilaiUTS.length - 1);
        int minUTS = UTSterendah(nilaiUTS, 0, nilaiUTS.length - 1);
        double avgUAS = RataUAS(nilaiUAS);

        System.out.println("Nilai UTS Tertinggi: " + maxUTS);
        System.out.println("Nilai UTS Terendah: " + minUTS);
        System.out.println("Rata-rata Nilai UAS: " + avgUAS);
    }
}
```

Hasil running:



https://github.com/AlfredaDhaifullah04/Semester-2/tree/master/Pertemuan_5