LAPORAN JOBSHEET 5 ALSD

1. Percobaan 1

Class Method

```java
package JOBSHEET5;

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

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

Class main

```java
package JOBSHEET5;
import java.util.Scanner;
public class mainfakktorial {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s:"Masukkan Nilai: ");
        int nilai = input.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));


    }
}
```

Run program

```
Masukkan Nilai: 5
Nilai Faktorial =5 menggunakan BF: 120
Nilai faktorial: 5 Menggunakan DC: 120
PS C:\Users\PC\Praktikum-ASD-1>
```

PERTANYAAN

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!

2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

JAWABAN

1. Pada logika if merupakan kondisi dasar atau base case jadi apabila n benar benar = 1 maka fungsi akan mengembalikan nilai 1 lalu untuk logika else ini merupakan recursive case yang mana metod akan memanggil dirinya sendiri sampai hasil nilai n =1
2. Ya pada method bf perulangan juga dapat menggunakan perulangan while

```
3   public class faktorial {
4       int faktorialBF(int n) {
5           int fakto = 1;
6           int i = 1;
7           while (i <= n) {
8               fakto *= i;
9               i++;
10          }
11          return fakto;
12      }
13
```

```
PROBLEMS 42   OUTPUT   DEBUG CONSOLE   TERMIN
TERMINAL

PS C:\Users\PC\Praktikum-ASD-1>  & 'C:\Prog
java.exe' '-XX:+ShowCodeDetailsInExceptionM
C\AppData\Roaming\Code\User\workspaceStorag
49b280ca\redhat.java\jdt_ws\Praktikum-ASD-1_
mainfakktorial'
Masukkan Nilai: 5
Nilai Faktorial =5 menggunakan BF: 120
Nilai faktorial: 5 Menggunakan DC: 120
PS C:\Users\PC\Praktikum-ASD-1>
```

3. Untuk perulangan fakto *= i; (Iteratif) sedangkan int fakto = n * faktorialDC(n-1); (Rekursif) Dengan Nilai fakto dikalikan i dalam loop sedangkan pada int fakto = n * faktorialDC(n-1); Nilai fakto dihitung dengan pemanggilan rekursif
4. metode iteratif lebih efisien dalam segi performa, sedangkan metode rekursif lebih elegan secara konsep dan cocok untuk proses divide and conquer.

2. Percobaan 2
   Class method

```java
package JOBSHEET5;

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));

            }
        }
    }
}
```

CLASS MAIN

```java
package JOBSHEET5;

import java.util.Scanner;

public class mainnpangkat {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print(s:"Masukkan Jumlah Elemen: ");
        int elemen = input.nextInt();

        pangkat[] png = new pangkat[elemen];

        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai basis elemen ke-" + (i + 1) + ": ");
            int basis = input.nextInt();
            System.out.print("Masukkan nilai pangkat elemen ke-" + (i + 1) + ": ");
            int pangkat = input.nextInt();

            png[i] = new pangkat(basis, pangkat);
        }

        System.out.println(x:"Hasil Pangkat Brute Force:");
        for (pangkat p : png) {
            System.out.println(p.nilai + "^" + p.Pangkat + " = " + p.pangkatBF(p.nilai, p.Pangkat));
        }

        System.out.println(x:"Hasil Pangkat Divide and Conquer:");
        for (pangkat p : png) {
            System.out.println(p.nilai + "^" + p.Pangkat + " = " + p.pangkatDC(p.nilai, p.Pangkat));
        }

        input.close();
    }
}
```

run program

```
Masukkan Jumlah Elemen: 3
Masukkan nilai basis elemen ke-1:
2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7
Hasil Pangkat Brute Force:
2^3 = 8
4^5 = 1024
6^7 = 279936
Hasil Pangkat Divide and Conquer:
2^3 = 8
4^5 = 1024
6^7 = 279936
PS C:\Users\PC\Praktikum-ASD-1>
```

PERTANYAAN

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!
3. Pada method pangkatBF()terdapat parameter untuk melewatkan nilai yang akan dipangkatkan dan pangkat berapa, padahal di sisi lain di class Pangkat telah ada atribut nilai dan pangkat, apakah menurut Anda method tersebut tetap relevan untuk memiliki parameter? Apakah bisa jika method tersebut dibuat dengan tanpa parameter? Jika bisa, seperti apa method pangkatBF() yang tanpa parameter?
4. Tarik kesimpulan tentang cara kerja method pangkatBF() dan pangkatDC()!

Jawaban

1.

pangkatBF() (Brute Force)

- Menggunakan looping (perulangan) untuk menghitung hasil pangkat.
- Kompleksitas waktu O(n) karena harus melakukan perkalian sebanyak n kali.

pangkatDC() (Divide and Conquer)

- Menggunakan teknik **rekursi** dengan membagi masalah menjadi submasalah yang lebih kecil.
- Kompleksitas waktu **O(log n)** karena hanya memerlukan sekitar log n pemanggilan rekursi.

2. a, tahap **combine** sudah termasuk dalam kode method pangkatDC().

```
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. ya method tersebut bias dibuat tanpa parameter

```
11      public int pangkatBF() {
12          int hasil = 1;
13          for (int i = 0; i < Pangkat; i++)
14              hasil *= nilai;
15          }
16          return hasil;
17      }
18
```

PROBLEMS 84 OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7

Hasil Pangkat Brute Force:
2^3 = 8
4^5 = 1024
6^7 = 279936

Hasil Pangkat Divide And Conquer:
2^3 = 8
4^5 = 1024
6^7 = 279936
PS C:\Users\PC\Praktikum-ASD-1>
```

4. **pangkatBF()** menggunakan perulangan untuk menghitung pangkat secara langsung, sementara **pangkatDC()** membagi masalah menjadi lebih kecil dengan rekursi, sehingga lebih efisien untuk pangkat besar.

3. Percobaan 3
   Class method

```java
package JOBSHEET5;

public class Sum {
    double keuntungan[];

    Sum(int el){
        keuntungan = new double[el];
    }

    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 rsum+lsum;
    }
}
```

Class main

```java
package JOBSHEET5;
import java.util.Scanner;
public class Mainsum {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s:"Masukkan Jumlah Elemen: ");
        int elemen = input.nextInt();

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

        System.out.println("Total keuntungan menggunakan bruteforce: " + sm.totalBF());
        System.out.println("Total keuntungan menggunakan divide and conquer: " + sm.totalDC(sm.keuntungan, 1:0, elemen-1));
        input.close();
    }
}
```

run program

```
Masukkan Jumlah Elemen: 5
Masukkan keuntungan ke-1: 10
Masukkan keuntungan ke-2: 20
Masukkan keuntungan ke-3: 30
Masukkan keuntungan ke-4: 40
Masukkan keuntungan ke-5: 50
Total keuntungan menggunakan bruteforce: 150.0
Total keuntungan menggunakan divide and conquer: 150.0
PS C:\Users\PC\Praktikum-ASD-1>
```

PERTANYAAN

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

```
double lsum = totalDC(arr, l, mid);
double rsum = totalDC(arr, mid+1, r);
```

3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?

```
return lsum+rsum;
```

4. Apakah base case dari `totalDC()`?
5. Tarik Kesimpulan tentang cara kerja `totalDC()`

JAWABAN

1. mid digunakan untuk membagi array menjadi dua bagian, sehingga algoritma Divide and Conquer dapat bekerja dengan membagi masalah menjadi submasalah yang lebih kecil.
2. Statement ini digunakan untuk menghitung total keuntungan di dua bagian array yang sudah dibagi oleh mid.

- lsum menghitung total keuntungan dari bagian kiri.
- rsum menghitung total keuntungan dari bagian kanan.

Metode ini memastikan bahwa setiap bagian dihitung secara rekursif sampai mencapai kondisi base case.

3. Penjumlahan ini diperlukan untuk menggabungkan hasil dari dua bagian array yang telah dihitung secara rekursif

4. Base case terjadi ketika hanya ada satu elemen yang tersisa dalam array, yaitu saat

l == r.

5. Metode totalDC() menggunakan pendekatan **Divide and Conquer** dengan cara membagi array menjadi dua bagian secara rekursif hingga mencapai satu elemen (base case). Kemudian, hasil dari kedua bagian dijumlahkan untuk mendapatkan total keseluruhan.

4.5 Latihan Praktikum

1. Sebuah kampus memiliki daftar nilai mahasiswa dengan data sesuai tabel di bawah ini

Nama	NIM	Tahun Masuk	Nilai UTS	Nilai UAS
Ahmad	220101001	2022	78	82
Budi	220101002	2022	85	88
Cindy	220101003	2021	90	87
Dian	220101004	2021	76	79
Eko	220101005	2023	92	95
Fajar	220101006	2020	88	85
Gina	220101007	2023	80	83
Hadi	220101008	2020	82	84

Tentukan:

a) Nilau UTS tertinggi tertinggi menggunakan Divide and Conquer!

b) Nilai UTS terendah menggunakan Divide and Conquer!

c) Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force!

JAWABAN

Method class

```java
package JOBSHEET5;

public class Mahasiswa {
    String nama;
    int uts, uas;

    Mahasiswa(String nama, int uts, int uas) {
        this.nama = nama;
        this.uts = uts;
        this.uas = uas;
    }

    int maxUTS(Mahasiswa[] mhs, int l, int r) {
        if (l == r) return mhs[l].uts;
        int mid = (l + r) / 2;
        int leftMax = maxUTS(mhs, l, mid);
        int rightMax = maxUTS(mhs, mid + 1, r);
        return Math.max(leftMax, rightMax);
    }

    int minUTS(Mahasiswa[] mhs, int l, int r) {
        if (l == r) return mhs[l].uts;
        int mid = (l + r) / 2;
        int leftMin = minUTS(mhs, l, mid);
        int rightMin = minUTS(mhs, mid + 1, r);
        return Math.min(leftMin, rightMin);
    }

    double avgUAS(Mahasiswa[] mhs) {
        double total = 0;
        for (Mahasiswa m : mhs) {
            total += m.uas;
        }
        return total / mhs.length;
    }
}
```

Main class

```java
package JOBSHEET5;

public class MainMahasiswa {
    Run | Debug
    public static void main(String[] args) {
        Mahasiswa[] mhs = {
            new Mahasiswa(nama:"Ahmad", uts:78, uas:82),
            new Mahasiswa(nama:"Budi", uts:85, uas:88),
            new Mahasiswa(nama:"Cindy", uts:90, uas:87),
            new Mahasiswa(nama:"Dian", uts:76, uas:79),
            new Mahasiswa(nama:"Eko", uts:92, uas:95),
            new Mahasiswa(nama:"Fajar", uts:88, uas:85),
            new Mahasiswa(nama:"Gina", uts:80, uas:83),
            new Mahasiswa(nama:"Hadi", uts:82, uas:84)
        };

        Mahasiswa mh = new Mahasiswa(nama:"Dummy", uts:0, uas:0);

        System.out.println("Nilai UTS tertinggi menggunakan Divide and Conquer: " + mh.maxUTS(mhs, l:0, mhs.length - 1));
        System.out.println("Nilai UTS terendah menggunakan Divide and Conquer: " + mh.minUTS(mhs, l:0, mhs.length - 1));
        System.out.println("Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force: " + mh.avgUAS(mhs));
    }
}
```

Run program

```
Nilai UTS tertinggi menggunakan Divide and Conquer: 92
Nilai UTS terendah menggunakan Divide and Conquer: 76
Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force: 85.375
```