Nama : Zaskia Maulidina Mutiara Hati

NIM : 244107060056

Kelas : SIB 1A

JOBSHEET 5 ALGORITMA STRUKTUR DATA

Percobaan 1

- **Class**

jobsheet_5 > Faktorial24.java > Faktorial24

```java
package jobsheet_5;

public class Faktorial24 {

    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;
        }
    }
}
```

- **Main**

jobsheet_5 > MainFaktorial24.java > MainFaktorial24

```java
package jobsheet_5;
import java.util.Scanner;
public class MainFaktorial24 {
    Run | Debug
    public static void main(String[] args) {
        Scanner input24 = new Scanner(System.in);
        System.out.print(s:"Masukkan nilai: ");
        int nilai = input24.nextInt();

        Faktorial24 fk = new Faktorial24();
        System.out.println("Nilai Faktorial " + nilai + " menggunakan BF: " + fk.faktorialBF(nilai));
        System.out.println("Nilai faktorial " + nilai + " menggunakan DC: " + fk.faktorialDC(nilai));
    }
}
```

- **Output**

```
Masukkan nilai: 5
Nilai Faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan DC: 120
```

Pertanyaan 1

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
    - Bagian if(n==1) berfungsi sebagai base case, yang menghentikan rekursi.
    - Bagian else berfungsi untuk menjalankan pemanggilan rekursif untuk menghitung factorial menggunakan Divide Conquer.

2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
    - Iya mungkin, bisa menggunakan while dan do while
    -
```
13  int faktorialBF_while(int n) {
14      int fakto = 1;
15      int i = 1;
16      while (i <= n) {
17          fakto = fakto * i;
18          i++;
19      }
20      return fakto;
21  }
```
    -
```
23  int faktorialBF_doWhile(int n) {
24      int fakto = 1;
25      int i = 1;
26      do {
27          fakto = fakto * i;
28          i++;
29      } while (i <= n);
30      return fakto;
31  }
```

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
    - fakto *= 1;
        - Mulai dari fakto = 1
        - Lalu kalikan fakto dengan 1, 2, 3, ..., sampai n
        - Hasil akhirnya dikembalikan

    - fakto = n * faktorialDC(n-1);
      Jika n = 1, langsung return 1 (berhenti di sini).
      Kalau n > 1, di pecah jadi lebih kecil:
        - 5! = 5 * faktorialDC(4)
        - 4! = 4 * faktorialDC(3)
        - 3! = 3 * faktorialDC(2)
        - 2! = 2 * faktorialDC(1)
        - 1! = 1 (base case)

      Setelah mencapai 1, hasilnya dikembalikan ke atas.

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
    - fakto *= 1; menggunakan cara iterative (perulangan)
    - fakto = n * faktorialDC(n-1); menggunakan rekursif (fungsi yang memanggil dirinya sendiri)

Percobaan 2

- **Class**

jobsheet_5 > Pangkat24.java > Pangkat24

```java
package jobsheet_5;

public class Pangkat24 {
    int nilai;
    int pangkat;

    Pangkat24(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));
            }
        }
    }

}
```

- **Main**

jobsheet_5 > MainPangkat24.java > ...

```java
package jobsheet_5;
import java.util.Scanner;
public class MainPangkat24 {
    Run | Debug
    public static void main(String[] args) {
        Scanner input24 = new Scanner(System.in);
        System.out.print(s:"Masukkan julah elemen: ");
        int elemen = input24.nextInt();

        Pangkat24[] png = new Pangkat24[elemen];
        for(int i=0; i<elemen; i++){
            System.out.print("Masukkan nilai basis elemen ke-" + (i+1) + ": ");
            int basis = input24.nextInt();
            System.out.print("Masukkan nilai pangkat elemen ke-" + (i+1) + ": ");
            int pangkat = input24.nextInt();
            png[i] = new Pangkat24(basis, pangkat);
        }

        System.out.println(x:"HASIL PANGKAT BRUTEFORCE");
        for (Pangkat24 p : png){
            System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatBF(p.nilai, p.pangkat));
        }
        System.out.println(x:"HASIL PANGKAT DIVIDE CONQUER");
        for (Pangkat24 p : png){
            System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatDC(p.nilai, p.pangkat));
        }
    }
}
```

- **Output**

```
Masukkan julah 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 BRUTEFORCE
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVIDE CONQUER
2^3: 8
4^5: 1024
6^7: 279936
```

Pertanyaan 2

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
   - PangkatBF(), menggunakan metode iteratif dengan looping
   - Cara Kerja :
     - Menginisialisasi hasil = 1.
     - Mengalikan hasil dengan a sebanyak n kali dalam looping.

   - PangkatDC(), menggunakan rekursif (fungsi yang memanggil dirinya sendiri)
   - Cara Kerja :
     - Jika n == 1, maka langsung mengembalikan a.
     - Jika n **genap**, hasil dihitung sebagai (pangkatDC(a, n/2) * pangkatDC(a, n/2)).
     - Jika n **ganjil**, hasil dihitung sebagai (pangkatDC(a, n/2) * pangkatDC(a, n/2) * a).

2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!
   - `return (pangkatDC(a, n/2)* pangkatDC(a, n/2)*a);` untuk n ganjil.
   - `return (pangkatDC(a, n/2)* pangkatDC(a, n/2));` untuk n genap.

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?
    - Iya, method tetap akan relevan untuk memiliki parameter.
    - Bisa,

```
20      int pangkatBF(){
21          int hasil = 1;
22          for(int i =0; i < pangkat; i++){
23              hasil = hasil * nilai;
24          }
25          return hasil;
26      }
```

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
    - Cara kerja pangkatBF() :
        - Menginisialisasi hasil = 1.
        - Mengalikan hasil dengan a sebanyak n kali dalam looping.
    - Contoh eksekusi pangkatBF(2, 5):
        - hasil = 1
        - Iterasi ke-1: hasil = 1 * 2 = 2
        - Iterasi ke-2: hasil = 2 * 2 = 4
        - Iterasi ke-3: hasil = 4 * 2 = 8
        - Iterasi ke-4: hasil = 8 * 2 = 16
        - Iterasi ke-5: hasil = 16 * 2 = 32
        - Hasil akhir: 32

    - Cara kerja pangkatDC() :
        - Jika n == 1, maka langsung mengembalikan a.
        - Jika n **genap**, hasil dihitung sebagai (pangkatDC(a, n/2) * pangkatDC(a, n/2)).
        - Jika n **ganjil**, hasil dihitung sebagai (pangkatDC(a, n/2) * pangkatDC(a, n/2) * a).
    - Contoh eksekusi pangkatDC(2, 5):
        - pangkatDC(2,5) → pangkatDC(2,2) * pangkatDC(2,2) * 2
        - pangkatDC(2,2) → pangkatDC(2,1) * pangkatDC(2,1)
        - pangkatDC(2,1) = 2
        - pangkatDC(2,2) = 2 * 2 = 4
        - pangkatDC(2,5) = 4 * 4 * 2 = 32

Percobaan 3

- **Class**

jobsheet_5 > Sum24.java > Sum24

```java
package jobsheet_5;

public class Sum24 {
    double keuntungan[];

    Sum24(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 lsum + rsum;
    }
}
```

- **Main**

jobsheet_5 > MainSum24.java > MainSum24

```java
package jobsheet_5;
import java.util.Scanner;
public class MainSum24 {
    Run | Debug
    public static void main(String[] args) {
        Scanner input24 = new Scanner(System.in);

        System.out.print(s:"Masukkan jumlah elemen: ");
        int elemen = input24.nextInt();

        Sum24 sm = new Sum24(elemen);
        for(int i = 0; i < elemen; i++){
            System.out.print("Masukkan Keuntungan ke-" + (i+1) + ": ");
            sm.keuntungan[i] = input24.nextDouble();
        }

        System.out.println("Total Keuntungan menggunakan Bruteforce: " + sm.totalBF());
        System.out.println("Total Keuntungan menggunakan Divide and Conquer: " + sm.totalDC(sm.keuntungan, l:0, sm.keuntungan.length-1));
    }
}
```

- **Output**

```
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
```

Pertanyaan 3

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
   - Variabel mid pada method TotalDC() digunakan untuk menentukan titik tengah array sehingga array dapat dibagi menjadi dua bagian.

2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

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

   - Digunakan untuk menghitung jumlah elemen di bagian kiri dan kanan dari array dengan cara memanggil fungsi totalDC() lagi (rekursi).
     - lsum = totalDC(arr, l, mid); → menghitung total dari bagian kiri (dari indeks l sampai mid).
     - rsum = totalDC(arr, mid + 1, r); → menghitung total dari bagian kanan (dari indeks mid + 1 sampai r).

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

```
return lsum+rsum;
```

   - Kode return lsum + rsum; diperlukan untuk menghitung hasil akhir dari seluruh array dengan cara menjumlahkan hasil dari bagian kiri (lsum) dan bagian kanan (rsum) yang sudah dihitung melalui rekursi.

4. Apakah base case dari totalDC()?
   -
```
if(l==r){
    return arr[l];
}
```

5. Tarik Kesimpulan tentang cara kerja totalDC()
   - Fungsi totalDC() menggunakan metode **Divide and Conquer** untuk menghitung total elemen dalam array dengan cara:
     - **Membagi (Divide)** → Array dibagi menjadi dua bagian menggunakan variabel mid.
     - **Menyelesaikan (Conquer)** → Fungsi dipanggil secara rekursif untuk menghitung jumlah elemen di bagian kiri dan kanan.
     - **Menggabungkan (Combine)** → Setelah hasil dari kedua bagian dihitung, nilainya dijumlahkan (return lsum + rsum).
     - **Basis Kasus** → Jika hanya ada satu elemen (l == r), fungsi langsung mengembalikan nilai elemen tersebut.

Latihan Praktikum

- **Class**

jobsheet_5 > MahasiswaNilai24.java > MahasiswaNilai24

```java
package jobsheet_5;

public class MahasiswaNilai24 {
    int[] nilaiUTS;
    int[] nilaiUAS;

    public MahasiswaNilai24(int[] uts, int[] uas) {
        nilaiUTS = uts;
        nilaiUAS = uas;
    }

    public int cariMaksimum(int kiri, int kanan) {
        if (kiri == kanan) return nilaiUTS[kiri];
        int tengah = (kiri + kanan) / 2;
        int maxKiri = cariMaksimum(kiri, tengah);
        int maxKanan = cariMaksimum(tengah + 1, kanan);
        return (maxKiri > maxKanan) ? maxKiri : maxKanan;
    }

    public int cariMinimum(int kiri, int kanan) {
        if (kiri == kanan) return nilaiUTS[kiri];
        int tengah = (kiri + kanan) / 2;
        int minKiri = cariMinimum(kiri, tengah);
        int minKanan = cariMinimum(tengah + 1, kanan);
        return (minKiri < minKanan) ? minKiri : minKanan;
    }
```

```java
    public double hitungRataRata() {
        int total = 0;
        for (int i = 0; i < nilaiUAS.length; i++) {
            total += nilaiUAS[i];
        }
        return (double) total / nilaiUAS.length;
    }
}
```

➢ **Main**

```java
package jobsheet_5;

public class MainMahasiswaNilai24 {
    Run | Debug
    public static void main(String[] args) {
        int[] uts = {78, 85, 90, 76, 79, 88, 80, 82};
        int[] uas = {82, 88, 87, 79, 95, 85, 83, 84};

        MahasiswaNilai24 mahasiswa = new MahasiswaNilai24(uts, uas);

        int nilaiTertinggi = mahasiswa.cariMaksimum(kiri:0, uts.length - 1);

        int nilaiTerendah = mahasiswa.cariMinimum(kiri:0, uts.length - 1);

        double rataRataUAS = mahasiswa.hitungRataRata();

        System.out.println(x:"===== HASIL PERHITUNGAN NILAI MAHASISWA =====");
        System.out.println("Nilai UTS Tertinggi : " + nilaiTertinggi);
        System.out.println("Nilai UTS Terendah  : " + nilaiTerendah);
        System.out.println("Rata-rata Nilai UAS : " + rataRataUAS);
    }
}
```

➢ **Output**

```
===== HASIL PERHITUNGAN NILAI MAHASISWA =====
Nilai UTS Tertinggi : 90
Nilai UTS Terendah  : 76
Rata-rata Nilai UAS : 85.375
```