LAPORAN HASIL PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

JOBSHEET 5

NAMA : MOHAMAT FAUZI ROHMAN

NIM : 244107020067

KELAS : 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

JOBSHEET 5

5. Praktikum

5.1 Percobaan 1: Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

5.1.1 Langkah-langkah Percobaan

1. Buatlah class baru dengan nama Faktorial
2. Lengkapi class Faktorial dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
    1. Tambahkan method faktorialBF():

```
public class Faktorial {

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

    2. Tambahkan method faktorialDC():

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

3. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.
    a. Di dalam fungsi main sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya

```
import java.util.Scanner;
public class MainFaktorial {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

    System.out.print("Masukkan nilai: ");
    int nilai = input.nextInt();
```

b. Kemudian buat objek dari class Faktorial dan tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()

```
 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));

    }
}
```

5.1.2 Verifikasi Hasil Percobaan

```
Masukkan nilai: 4
Nilai Faktorial 4 menggunakan BF: 24
Nilai Faktorial 4 menggunakan DC: 24
```

5.1.3 Pertanyaan

1. – IF adalah kondisi dasar yang menghentikan rekursi.
   – ELSE adalah bagian yang membagi masalah dan memanggil fungsi secara rekursif hingga mencapai kondisi dasar.
2. Bisa, menggunakan perulangan while

```
int faktorialBF(int n){
        int fakto = 1;
        while (n > 1) {
            fakto *= n;
            n -= 1;
        }
        return fakto;
    }
```

```
Masukkan nilai: 4
Nilai Faktorial 4 menggunakan BF: 24
Nilai Faktorial 4 menggunakan DC: 24
PS C:\Users\ASUS\OneDrive\Documents\
```

3. fakto *= i; ialah dimana nilai variabel fakto akan diperbarui dengan cara mengalikan fakto dengan i.
   Sedangkan fakto = n * faktorialDC(n-1); adalah mendeklarasikan variabel fakto dan menginisialisasinya dengan n dikalikan method faktorialDC(n-1).

4. – faktorialBF() menggunakan prinsip brute force, yang dimana menggunakan perulangan untuk menghitung hasil yang dicari
   – faktorialDC() menggunakan prinsip divide and conquer, yang menggunakan metode memecah masalah menjadi sub-masalah yang lebih kecil.

5.2 Percobaan 2 : Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

5.2.1 Langkah-langkah Percobaan

1. Buatlah class baru dengan nama Pangkat, dan di dalam class Pangkat tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya

```
public class Pangkat {

int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter. Pada class Pangkat tersebut, tambahkan method PangkatBF() dan method PangkatDC()

```
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)*2);
        }else{
            return (pangkatDC(a, n/2) * pangkatDC(a, n/2));

        }
        }
    }
}
```

3. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah elemen yang akan dihitung pangkatnya.

```
import java.util.Scanner;
public class MainPangkat {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

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

4. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.

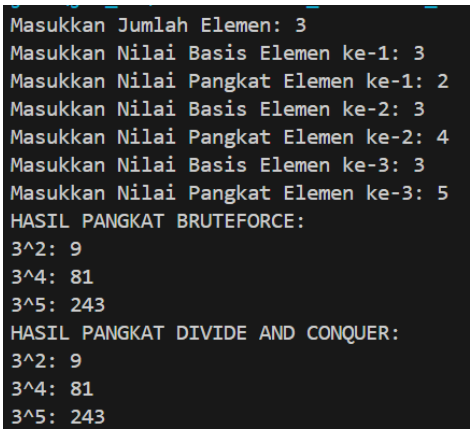
```
Pangkat[] png = new Pangkat[elemen];
  for(int i = 0; i < elemen; i++){
    System.out.print("Masukkan Nilai Basis Elemen ke-" + (i+1) + ": ");
    int basis = input.nextInt();
    System.out.print("Masukkan Nilai Pangkat Elemen ke-" + (i+1) + ": ");
    int pangkat = input.nextInt();
    png[i] = new Pangkat(basis, pangkat);
    }
```

5. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().

```
 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));

    }
  }
}
```

5.2.2 Verfikasi Hasil Percobaan



5.2.3 Pertanyaan

1. -- pangkatBF() menggunakan brute force untuk menghitung pangkat. Dengan cara melakukan perkalian berulang sebanyak n kali.
   -- pangkatDC() menggunakan prinsip divide and conquer dalam menghitung pangkat. Jika n ganjil, maka hasil pangkat dikuadratkan kemudian dikalikan dengan a. Jika n genap, kita dapat menghitung pangkat dengan cara mengkuadratkan hasil dari pangkat

2. Ya, combine terdapat pada 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 tetap relevan untuk memiliki parameter dan method tersebut bisa dibuat dengan tanpa parameter

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

4. PangkatBF();
   - Menginisialisasi variabel hasil dengan nilai 1
   - Menggunakan perulangan for untuk mengalikan hasil dengan nilai sebanyak pangkat kali
   - Setiap iterasi loop, hasil diperbarui dengan perkalian antara hasil dengan nilai

   PangkatDC();

   - Jika n=1, maka akan langsung dikembalikan ke a
   - Jika n ganjil, selanjutnya akan memanggil dirinya sendiri 2 kali dengan n/2 dan hasilnya dikalikan dengan a
   - Jika n genap, selanjutnya akan memanggil dirinya sendiri 2 kali dengan n/2 dan mengalikan hasil keduanya

5.3 Percobaan 3: Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

5.3.1 Langkah-langkah Percobaan

1. Buat class baru yaitu class Sum. Tambahkan pula konstruktor pada class Sum.

```
public class Sum {

double keuntungan[];

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

2. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara iterative. Dan tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer

```
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;
 }
}
```

3. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum

```
import java.util.Scanner;
public class MainSum {
   public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

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

4. Buat objek dari class Sum. Lakukan perulangan untuk mengambil input nilai keuntungan dan masukkan ke atribut keuntungan dari objek yang baru dibuat tersebut!

```
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();
    }
```

5. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)

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

  }
}
```

5.3.2 Verfikasi Hasil Percobaan

```
Masukkan Jumlah Elemen: 4
Masukkan Keuntungan ke-1: 20
Masukkan Keuntungan ke-2: 25
Masukkan Keuntungan ke-3: 15
Masukkan Keuntungan ke-4: 35
Total Keuntungan Menggunakan Bruteforce: 95.0
Total Keuntungan Menggunakan Divide and Conquer: 95.0
```

5.3.3 Pertanyaan

1. Variabel mid digunakan untuk menentukan titik tengah dari array. Dengan membagi array menjadi dua bagian (kiri dan kanan), kita dapat memecah masalah besar menjadi dua sub-masalah yang lebih kecil.
2. Untuk menghitung total dari elemen di bagian kiri (dari l hingga mid) dan bagian kanan (dari mid + 1 hingga r).
3. Digunakan untuk menggabungkan hasil dari dua sub-masalah yang telah dihitung secara terpisah.
4. Jika hanya terdapat satu elemen saja, maka dapat langsung dikembalikan nilai elemen tersebut

```
if (l == r) {
        return arr[l];
    }
```

5. Metode TotalDC() membagi masalah menjadi dua sub-masalah yang lebih kecil, Kemudian menghitung total dari dua bagian secara rekursif, Setelah mendapatkan total dari kedua bagian dijumlahkan kedua hasil tersebut untuk mendapatkan total keseluruhan.