LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA
BRUTE FORCE DAN DIVIDE CONQUEROR

ABIM MUSTAWA

244107020078

KELAS TI-1B

PRODI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

1. Percobaan Praktikum

1.1 Percobaan 1 (Nilai Faktorial)

1. Buat folder baru bernama Jobsheet5 di dalam repository Praktikum ASD
2. Buatlah class baru dengan nama Faktorial
3. Lengkapi class Faktorial dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
   a) Tambahkan method faktorialBF():

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

   b) Tambahkan method faktorialDC():

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

4. 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

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

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

```
Faktorial01 fk = new Faktorial01();
System.out.println("Nilai faktorial " + nilai + "menggunakan BF: " + fk.faktorialBF(nilai));
System.out.println("Nilai faktorial " + nilai + "menggunakan DC: " + fk.faktorialDC(nilai));
```

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

Pertanyaan:

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
   Jawab: Penggunaan if dan else memiliki peran penting dalam menentukan **kapan rekursi berhenti** (base case) dan **kapan rekursi dilanjutkan** (recursive case)

Bagian if:
Base Case: Ini adalah kondisi berhenti untuk rekursi
Bagian else:
Recursive Case: Ini adalah bagian di mana masalah dibagi menjadi sub-masalah yang lebih kecil. Rekursi akan terus berlanjut sampai mencapai base case (n=1)

2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
   Jawab: Memungkinkan, kita bisa menggunakan while

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

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
   Jawab:
   fakto *=
   - Ini adalah operasi perkalian yang dilakukan dalam **perulangan** (iteratif)
   - Digunakan dalam metode faktorialBF() untuk menghitung faktorial secara bertahap
   - Nilai fakto diperbarui secara bertahap dengan mengalikannya dengan i

   int fakto = n * faktorialDC(n-1);

   - Ini adalah operasi rekursif yang memecah masalah n! menjadi n×(n−1)!
   - Digunakan dalam metode faktorialDC() untuk menghitung faktorial dengan pendekatan Divide and Conquer
   - Nilai fakto dihitung dengan mengalikan n*n* dengan hasil rekursif dari (n−1)!

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
   Jawab:
   faktorialBF() :
   - **Pendekatan**: Iteratif (menggunakan perulangan for, while, atau do-while).
   - Menghitung faktorial secara bertahap dengan mengalikan nilai dari 1 hingga nn.
   - Tidak memecah masalah menjadi sub-masalah

   faktorialDC() :
   - **Pendekatan**: Rekursif (menggunakan pemanggilan fungsi diri sendiri).
   - Memecah masalah n!n! menjadi sub-masalah n×(n−1)!n×(n−1)!.
   - Menggunakan base case untuk menghentikan rekursi

1.2 Percobaan 2 (Hasil Pangkat)

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

```
int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter

```
int nilai, pangkat;
pangkat01(int n, int p){
    nilai = n;
    pangkat = p;
}
```

3. Pada class Pangkat tersebut, tambahkan method PangkatBF()

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

4. Pada class Pangkat juga tambahkan method PangkatDC()

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

5. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
6. 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.

```
Scanner input = new Scanner(System.in);
System.out.print(s:"Masukkan jumlah elemen: ");
int elemen = input.nextInt();
```

7. 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.

```
pangkat01[] png = new pangkat01[elemen];
for (int i = 0; i < elemen; i++) {
    System.out.print("Masukkan basis elemen ke-" +(i+1)+": ");
    int basis = input.nextInt();
    System.out.print("Masukkan pangkat elemen ke-" +(i+1)+": ");
    int pangkat = input.nextInt();
    png[i] = new pangkat01(basis, pangkat);
}
```

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

```
System.out.println(x:"HASIL PANGKAT BRUTEFORCE: ");
for (pangkat01 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 (pangkat01 p : png) {
    System.out.println(p.nilai+"^"+p.pangkat+": " + p.pangkatDC(p.nilai, p.pangkat));
}
```

```
Masukkan jumlah elemen: 3
Masukkan basis elemen ke-1: 2
Masukkan pangkat elemen ke-1: 3
Masukkan basis elemen ke-2: 4
Masukkan pangkat elemen ke-2: 5
Masukkan basis elemen ke-3: 6
Masukkan pangkat elemen ke-3: 7
HASIL PANGKAT BRUTEFORCE:
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVIDE AND CONQUER:
2^3: 8
4^5: 1024
6^7: 279936
```

Pertanyaan:

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
   Jawab:
   PangkatBF():
   - **Pendekatan**: Iteratif (menggunakan perulangan).
   - Menghitung pangkat dengan mengalikan bilangan a*a* sebanyak n*n* kali.
   - Contoh: 2^3 = 2 × 2 × 2 = 8

   PangkatDC():
   - **Pendekatan**: Rekursif (menggunakan pemanggilan fungsi diri sendiri).
   - Memecah masalah an*an* menjadi sub-masalah yang lebih kecil.
   - Jika n genap, a^n = a^n/2 × a^n/2
   - Jika n ganjil, a^n = a^n/2 × a^n/2 × *a*

2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!
   Jawab:
   Sudah, tahap **combine** sudah termasuk dalam kode pangkatDC(). Tahap combine adalah tahap di mana hasil dari sub-masalah digabungkan untuk menghasilkan solusi akhir

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

Jika n genap: Hasil dari dua pemanggilan rekursif a^n/2 dikalikan untuk menghasilkan a^n
Jika n ganjil: Hasil dari dua pemanggilan rekursif a^n/2 dikalikan, kemudian dikalikan lagi dengan a untuk menghasilkan a^n

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?
   Jawab:
   method **pangkatBF()** tetap relevan memiliki parameter karena:
   - Parameter memungkinkan method untuk digunakan secara fleksibel, tidak terikat pada nilai atribut class.
   - Method ini bisa digunakan untuk menghitung pangkat dari bilangan dan pangkat apa pun, tidak hanya yang disimpan dalam atribut class.

   method **pangkatBF()** bisa jika tanpa menggunakan parameter:

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

```
System.out.println(x:"HASIL PANGKAT BRUTEFORCE: ");
for (pangkat01 p : png) {
    System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatBF());
}
```

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
   Jawab:
   pangkatBF()
   - **Inisialisasi**: Variabel hasil diatur ke 1.
   - **Perulangan**: Lakukan perulangan sebanyak n*n* kali:
     - Pada setiap iterasi, kalikan hasil dengan a.
   - **Hasil**: Kembalikan nilai hasil.

   Contoh 2^3:

   Iterasi 1: hasil=1×2=2*h*

   Iterasi 2: hasil=2×2=4

   Iterasi 3: hasil=4×2=8

   Hasil: 8

pangkatDC()

- **Base Case**: Jika n=1n=1, kembalikan aa.
- **Recursive Case:**
  - Jika n genap: a^n = a^n/2 × a^n/2 = a^n/2
  - Jika n ganjil: a^n = a^n/2 × a^n/2 = a^n/2 × a
- **Combine**: Gabungkan hasil dari sub-masalah

Contoh: 2^4:

2^4 = 2^2 × 2^2

2^2 = 2^1 × 2^1

2^1 = 2 (base case).

Hasil: 2 × 2 = 4, lalu 4 × 4 = 16.

1.3 Percobaan 3 (Sum Array)

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

```
double keuntungan[];

Sum01(int el){
    keuntungan = new double[el];
}
```

2. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara iterative.

```
double totalBF(){
    double total =0;
    for (int i = 0; i < keuntungan.length; i++)
        total = total + keuntungan[i];
    }
    return total;
}
```

3. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer

```
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;
}
```

4. 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

```
Scanner input = new Scanner(System.in);
System.out.print(s:"Masukkan jumlah elemen: " );
int elemen = input.nextInt();
```

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

```
Sum01 sm = new Sum01(elemen);
for (int i = 0; i < elemen; i++) {
    System.out.print("Masukkan keuntungan ke-" +(i+1) + ": ");
    sm.keuntungan[i] = input.nextDouble();
}
```

6. 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 Devine and Conquer: " + sm.totalDC(sm.keuntungan, l:0, elemen-1));
```

```
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 Devine and Conquer: 150.0
```

Pertanyaan:

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
   Jawab: Untuk titik awal dari penghitungan sebelah kanan, karena pada method ini perhitungan dibagi menjadi 2

2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?
   Jawab: lsum untuk menghitung jumlah nilai mulai dari indeks paling awal ke mid, lalu untuk rsum untuk menghitung jumlah nilai dari indeks Tengah + 1 ke indeks ukuran array-1

3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?
   Jawab: Karena penilaiannya dibagi menjadi 2, yaitu kanan dan kiri

4. Apakah base case dari totalDC()?
   Jawab: if l == r

5. Tarik Kesimpulan tentang cara kerja totalDC()
   Jawab: Perhitungan keuntungan menggunakan fungsi rekursif untuk melakukan proses divide yang diimplementasikan dengan pemiihan (if-else, if-else), kemudian melakukan tahapan conquer untuk menyelesaikan setiap masalah tersebut yang dibagi menjadi 2 bagian yaitu kanan dan kiri, lalu pada tahap akhir atau combine maka semua hasil penyelesaian tadi dijadikan satu menjadi solusi