LAPORAN PRAKTIKUM ALGORITMA DASAR PEMROGRAMAN

JOBSHEET 5

SHAFIQA NABILA MAHARANI KHOIRUNNISA

244107020221

TI – 1B

5.2

```
Masukkan Nilai: 5
Nilai Faktorial 5 Menggunakan BF : 120
Nilai Faktorial 5 Menggunakan DC : 120
PS C:\Users\fika\MATKUL SEMESTER 2\praktikum-ASD\Jobsheet 5>
```

PERTANYAAN :

1. **Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!**
   - if (n == 0) return 1; → Ini adalah base case yang menghentikan rekursi dan mengembalikan nilai faktorial dari 0, yaitu 1.
   - else return n * faktorialDC(n - 1); → Ini adalah recursive case yang memanggil dirinya sendiri dengan nilai n-1 hingga mencapai base case.

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

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

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
   - fakto *= i; digunakan dalam iterasi (Brute Force), di mana faktorial dihitung dengan perkalian dalam loop.
   - int fakto = n * faktorialDC(n-1); digunakan dalam rekursi (Divide and Conquer), di mana faktorial dihitung dengan memanggil dirinya sendiri sampai mencapai base case.

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
   - faktorialBF() menggunakan pendekatan iteratif, lebih mudah dipahami tetapi membutuhkan lebih banyak operasi dalam loop.

- faktorialDC() menggunakan pendekatan rekursif, lebih efisien dalam konsep tetapi menggunakan lebih banyak memori karena pemanggilan rekursif.

5.3

```
Masukkan Jumlah Elemen : 3
Masukkan Nilai Baris Elemen ke-1 : 2
Masukkan Nilai Pangkat Elemen ke-1 : 3
Masukkan Nilai Baris Elemen ke-2 : 4
Masukkan Nilai Pangkat Elemen ke-2 : 5
Masukkan Nilai Baris 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 AND CONQUER
2^3: 8
4^5: 1024
6^7: 279936
```

PERTANYAAN :

1. **Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!**
   pangkatBF() menggunakan iterasi dengan kompleksitas O(n), sedangkan pangkatDC() menggunakan rekursi dengan kompleksitas O(log n), sehingga lebih efisien untuk pangkat besar.

2. **Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!**
   ```
   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 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?**

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

4. **Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!**
   pangkatDC() lebih efisien dibandingkan pangkatBF(), terutama untuk pangkat besar karena memiliki kompleksitas O(log n) dibandingkan O(n) pada brute force.

5.4

```
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\fika\MATKUL SEMESTER 2\praktikum-ASD\Jobsheet 5>
```

PERTANYAAN :

1. **Kenapa dibutuhkan variable mid pada method TotalDC()?**
   mid digunakan untuk membagi array menjadi dua bagian agar bisa dihitung secara rekursif dengan metode Divide and Conquer.

2. **Untuk apakah statement di bawah ini dilakukan dalam TotalDC()? Algoritma dan Struktur Data 2024-2025 – D4 Teknik Informatika 7 Tim Ajar Algoritma dan Struktur Data 2024-2025 – D4 Teknik Informatika Jurusan Teknologi Informasi-Politeknik Negeri Malang**
   digunakan untuk menghitung total dari bagian kiri (lsum) dan kanan (rsum) secara rekursif sebelum digabungkan.

3. **Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?**
   Base case terjadi saat hanya ada satu elemen (l == r), sehingga langsung dikembalikan nilainya tanpa rekursi lebih lanjut.

4. **Apakah base case dari totalDC()?**
   totalDC() membagi array menjadi dua bagian, menghitung jumlah masing-masing bagian secara rekursif, lalu menggabungkan hasilnya, membuatnya lebih efisien dibandingkan iterasi pada data besar.