LAPORAN PRAKTIKUM
JOBSHEET 5

VANESA MARDIANA PUTRI
244107020129 / 23
KELAS TI 1B

PRODI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

1. **Percobaan 1**

```
Masukkan nilai:
5
Nilai Faktorial 5 Menggunakan BF : 120
Nilai Faktorial 5 Menggunakan DC 120
PS D:\kuliah\smt 2\alsd\praktikum-alsd-1>
```

Jawaban :

1. If menangani kondisi dasar untuk menghentikan rekursi, sedangkan else menangani pemecahan masalah menjadi sub-masalah yang lebih kecil.
2. Mungkin, berikut pembuktiannya :

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

3. fakto *= i digunakan dalam iterasi (Brute Force), di mana faktorial dihitung dengan perkalian dalam loop. Sementara 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. Kesimpulan : gunakan faktorialBF() jika ingin lebih efisien dan cepat, terutama untuk angka besar dan gunakan faktorialDC() jika ingin kode lebih simple dan elegan, tetapi hati-hati penggunaan memori yang lebih besar.

2. **Percobaan 2**

```
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
PS D:\kuliah\smt 2\alsd\praktikum-alsd-1>
```

Jawaban pertanyaan :

1. Gunakan faktorialBF() jika ingin lebih efisien dan cepat, terutama untuk angka besar dan gunakan faktorialDC() jika ingin kode lebih simple dan elegan, tetapi hati-hati penggunaan memori yang lebih besar.
2.
```
return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
```
3.
```
int pangkatBF(){
    int hasil = 1;
    for (int i = 0; i < pangkat; i++) {
        hasil *= nilai;
    }
    return hasil;
}
```
4. pangkatBF() (Iteratif) : Menggunakan perulangan untuk menghitung pangkat dengan cara perkalian berulang. Mudah dipahami, tetapi lambat jika n besar.
   pangkatDC() (Rekursif) : Menggunakan rekursi dan membagi n menjadi lebih kecil (n/2) untuk mengurangi jumlah perkalian. Lebih cepat untuk angka besar, tetapi menggunakan lebih banyak memori.
   Gunakan pangkatBF() jika n kecil dan ingin cara sederhana.
   Gunakan pangkatDC() jika n besar dan butuh efisiensi tinggi.

3. Percobaan 3

```
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 D:\kuliah\smt 2\alsd\praktikum-alsd-1>
```

Jawaban pertanyaan :

1. Mid penting untuk membagi array secara efisien dan memungkinkan rekursi berjalan dengan benar.
2. Statement ini membantu membagi array dan menghitung totalnya secara rekursif, mengikuti metode Divide and Conquer agar lebih efisien.
3. Penjumlahan ini menggabungkan hasil dari dua bagian array yang telah dihitung secara rekursif, sehingga kita mendapatkan total seluruh elemen array secara akurat.
4.
```
if (l==r) {
    return arr[l];
}
```

5. Metode ini membagi array menjadi dua bagian lebih kecil, menghitung jumlahnya secara rekursif, lalu menggabungkan hasilnya untuk mendapatkan total keseluruhan array.