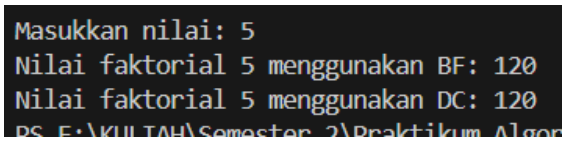
Nama : Wahyudi Satriawan Hamid

Nim : 244107020137

Verifikasi Hasil Percobaan 1



Pertanyaan 1

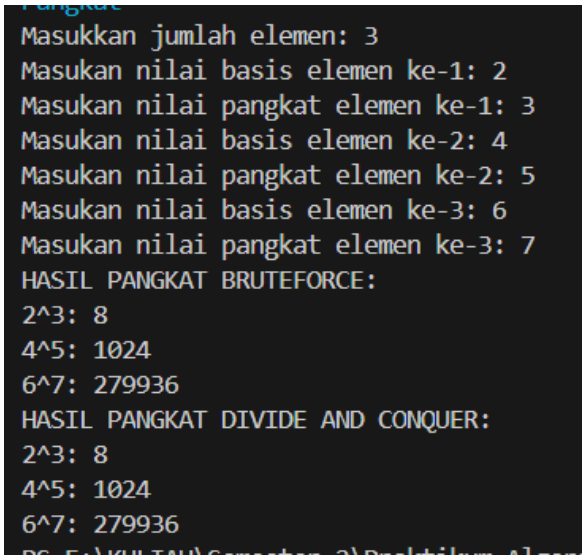
1. Perbedaan penggunaan `if` dan `else` dalam `faktorialDC()`
   - `if (n == 1)` → Base case, menghentikan rekursi dengan mengembalikan `1`.
   - `else` → Recursive case, memanggil fungsi sendiri (`n * faktorialDC(n-1)`) hingga mencapai base case.
2. Ya, kita bisa mengganti perulangan **for** dengan perulangan **while** atau **rekursi**.

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

3. Perbedaan utama:
   - `fakto *= i;` digunakan dalam perulangan iteratif.
   - `int fakto = n * faktorialDC(n-1);` digunakan dalam rekursi.

4. Kesimpulan:
   - Brute Force (faktorialBF) cocok untuk perhitungan langsung dan efisien dalam penggunaan memori.
   - Divide & Conquer (faktorialDC) lebih elegan tetapi kurang efisien untuk angka besar karena banyak pemanggilan rekursif.

Verifikasi Hasil Percobaan 2



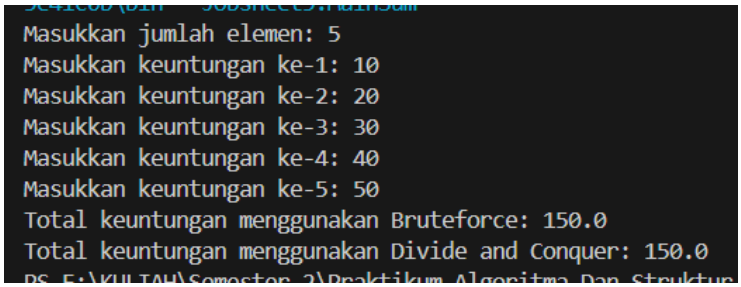
Pertanyaan 2

1. Perbedaan `pangkatBF()` dan `pangkatDC()`
   - `pangkatBF()` (Brute Force): Menghitung pangkat dengan perulangan `for`, mengalikan nilai secara langsung sebanyak `n` kali.
   - `pangkatDC()` (Divide and Conquer): Membagi masalah menjadi lebih kecil dengan rekursi, mengurangi jumlah perkalian dengan cara membagi `n` menjadi dua.
2.
```
int temp = pangkatDC(a, n / 2);
if (n % 2 == 0) {
     return temp * temp;  // Combine untuk n genap
} else {
     return temp * temp * a;  // Combine untuk n ganjil
}
```

3. Apakah `pangkatBF()` tetap relevan dengan parameter?
   - Dengan parameter → Lebih fleksibel, bisa digunakan untuk menghitung pangkat dari nilai lain yang berbeda.
   - Tanpa parameter → Memanfaatkan atribut yang sudah ada (`nilai` dan `pangkat`), tidak perlu meneruskan nilai saat pemanggilan.

4. Cara Kerja `pangkatBF()` vs `pangkatDC()`
    - `pangkatBF()`
        a. Menggunakan perulangan `for` untuk mengalikan nilai `n` kali.
        b. Kompleksitas O(n) (bertambah linear seiring bertambahnya `n`).
    - `pangkatDC()`
        a. Membagi pangkat menjadi bagian lebih kecil (`n/2`).
        b. Memproses rekursi hingga mencapai pangkat terkecil.
        c. Menggabungkan hasil untuk mendapatkan pangkat yang diminta.
        d. Kompleksitas O(log n) (jauh lebih cepat karena membagi masalah menjadi setengah setiap langkah).

Verifikasi Hasil Percobaan 3



Pertanyaan 3

1. Variabel mid digunakan untuk membagi array menjadi dua bagian, yaitu:
    - Bagian kiri → dari indeks l hingga mid.
    - Bagian kanan → dari indeks mid + 1 hingga r.
2. Statement ini digunakan untuk menghitung total keuntungan pada bagian kiri dan kanan array secara rekursif.
    - `lsum = totalDC(arr, l, mid);`
      Menghitung total keuntungan dari bagian kiri array.
    - `rsum = totalDC(arr, mid+1, r);`
      Menghitung total keuntungan dari bagian kanan array.

    Metode ini bekerja secara rekursif, sehingga setiap bagian akan terus dibagi lagi hingga hanya tersisa satu elemen (base case).

3. Penjumlahan ini dilakukan karena:
    - Setelah array dibagi menjadi dua bagian, masing-masing dihitung totalnya.
    - Untuk mendapatkan total keseluruhan, hasil dari kedua bagian harus dijumlahkan.
    - Ini merupakan tahap "Combine" dalam Divide and Conquer, di mana solusi dari sub-masalah digabung untuk mendapatkan solusi akhir.
4. Base case terjadi ketika hanya ada **satu elemen yang tersisa** di dalam array:

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

5. **Kesimpulan tentang cara kerja `totalDC()`**
    - **Divide (Pemisahan)**

      Array dibagi menjadi dua bagian menggunakan `mid`.
    - **Conquer (Penyelesaian Sub-masalah)**

      Masing-masing bagian dihitung totalnya menggunakan rekursi hingga hanya tersisa satu elemen.
    - **Combine (Penggabungan Solusi)**

      Hasil dari bagian kiri (`lsum`) dan bagian kanan (`rsum`) dijumlahkan untuk mendapatkan hasil akhir.