LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

Dosen Pengajar : Triana Fatmawati, S.T, M.T

Jobsheet-4 BRUTE FORCE DAN DIVIDE CONQUER

Nama : Surya Rahmat Fatahillah

NIM : 2341760020

Prodi : Sistem Informasi Bisnis

JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG 2023/2024

4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

4.2.1 Langkah-langkah Percobaan

Buat Project baru, dengan nama “BruteForceDivideConquer”. Buat package dengan nama minggu5. Buatlah class baru dengan nama Faktorial. Berikut adalah class Faktorial

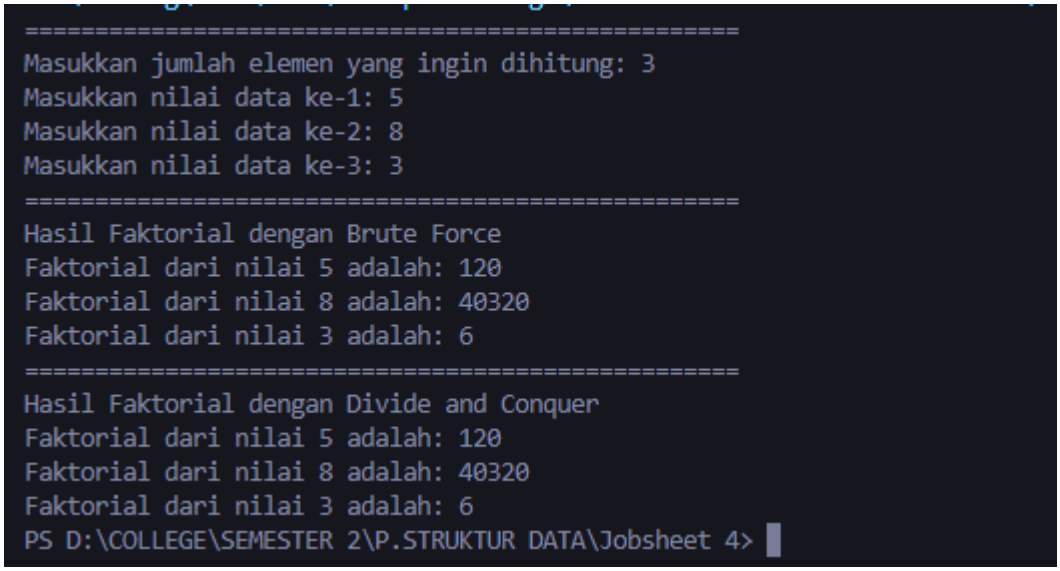
```java
package BruteForceDivideConquer.minggu5;
public class Faktorial27 {
    public int nilai;

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

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

```

.Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial. Berikut adalah class MainFaktorial:

```java
package BruteForceDivideConquer.minggu5;

import java.util.Scanner;

public class MainFaktorial27 {

    public class MainFaktorial03 {
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.println("================================================");
            System.out.print("Masukkan jumlah elemen yang ingin dihitung: ");
            int elemen = sc.nextInt();

            Faktorial27[] fk = new Faktorial27[elemen];
            for (int i = 0; i < elemen; i++) {
                fk[i] = new Faktorial27();
                System.out.print("Masukkan nilai data ke-" + (i + 1) + ": ");
                fk[i].nilai = sc.nextInt();
            }

            System.out.println("================================================");
            System.out.println("Hasil Faktorial dengan Brute Force");
            for (int i = 0; i < elemen; i++) {
                System.out
                        .println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + fk[i].faktorialBF(fk[i].nilai));
            }
            System.out.println("================================================");
            System.out.println("Hasil Faktorial dengan Divide and Conquer");
            for (int i = 0; i < elemen; i++) {
                System.out
                        .println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + fk[i].faktorialDC(fk[i].nilai));
            }
        }
    }

}
```

4.2.2 Verifikasi Hasil Percobaan Cocokkan hasil

compile kode program anda dengan gambar berikut ini.



4.2.3 Pertanyaan

1. Jelaskan mengenai base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial!
2. Pada implementasi Algoritma Divide and Conquer Faktorial apakah lengkap terdiri dari 3 tahapa divide, conquer, combine? Jelaskan masing-masing bagiannya pada kode program!
3. Apakah memungkinkan perulangan pada method faktorialBF() dirubah selain menggunakan for?Buktikan!
4. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!
5. Buktikan dengan inputan elemen yang di atas 20 angka, apakah ada perbedaan waktu eksekusi?

Jawaban!

1. Base case dalam method faktorialDC adalah ketika parameter n sama dengan 1. Ini diperiksa menggunakan pernyataan if(n == 1). Ketika n sama dengan 1, maka nilai faktorial adalah 1. Oleh karena itu, method mengembalikan nilai 1. ika parameter n tidak sama dengan 1, algoritma akan menggunakan pendekatan rekursif. Dalam rekursi, nilai faktorial dari n diperoleh dengan mengalikan n dengan nilai faktorial dari (n-1). Hal ini direpresentasikan oleh pernyataan int fakto = n * faktorialDC(n-1);. Masalah yang lebih besar (mencari faktorial dari n) dipisahkan menjadi masalah yang lebih kecil (mencari faktorial dari n-1). Setiap iterasi dari rekursi, n dikurangi 1, sehingga kita bergerak menuju base case. Setelah mencapai base case (ketika n sama dengan 1), hasil dari setiap langkah rekursi dikalikan satu sama lain untuk mendapatkan hasil akhir. Secara spesifik, hasil faktorial dari n adalah hasil perkalian dari n dengan nilai faktorial dari (n-1), dan seterusnya hingga mencapai base case.

2. Pada implementasi Algoritma Divide and Conquer di method faktorialDC Faktorial tidak lengkap terdiri dari 3 tahapan. Pada method **faktorialDC**, hanya terdapat dua tahap: **Divide:** Membagi masalah dengan mengecek nilai n. Jika n sama dengan 1, maka langsung direkursif if(n == 1){

   return 1;

   }else{

   ...}

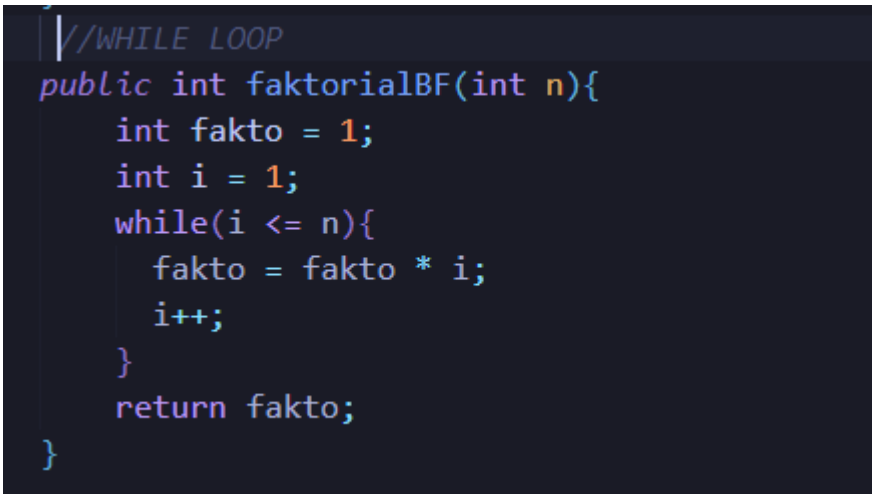
   Conquer: Menyelesaikan sub-masalah dengan mengalikan n dengan hasil rekursif faktorialDC(n-1).

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

   Namun, tahap **penggabungan (Combine)** tidak secara eksplisit ada karena pada kasus faktorial, hasil dari sub-masalah rekursif langsung dikalikan dengan n tanpa perlu proses penggabungan tambahan. Ini karena struktur dari masalah faktorial itu sendiri, di mana hasil akhir dari setiap sub-masalah sudah merupakan bagian dari solusi akhir, sehingga tidak memerlukan tahap penggabungan yang terpisah.

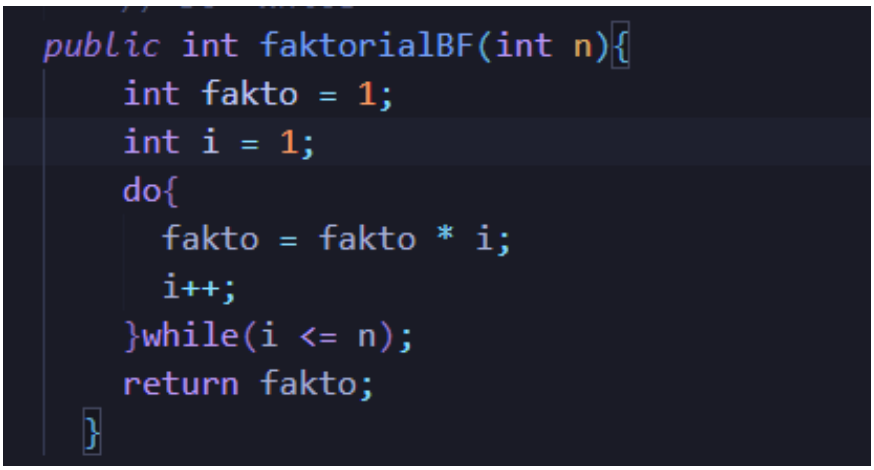
3. Ya, perulangan pada method faktorialBF() memungkinkan dirubah selain menggunakan for. Berikut beberapa contohnya:

   - Menggunakan While Loop:



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

   - Menggunakan Do While:



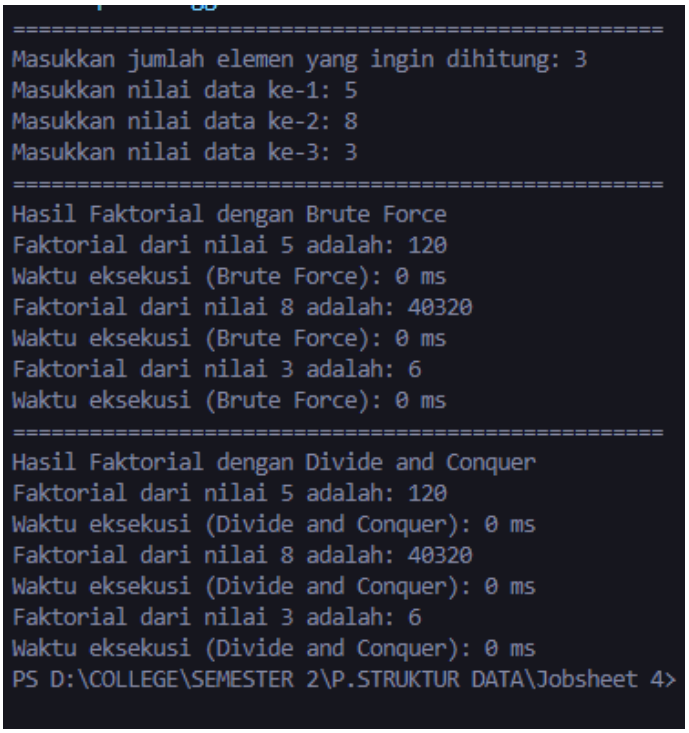
```
public int faktorialBF(int n){
    int fakto = 1;
    int i = 1;
    do{
      fakto = fakto * i;
      i++;
    }while(i <= n);
    return fakto;
  }
```

   -

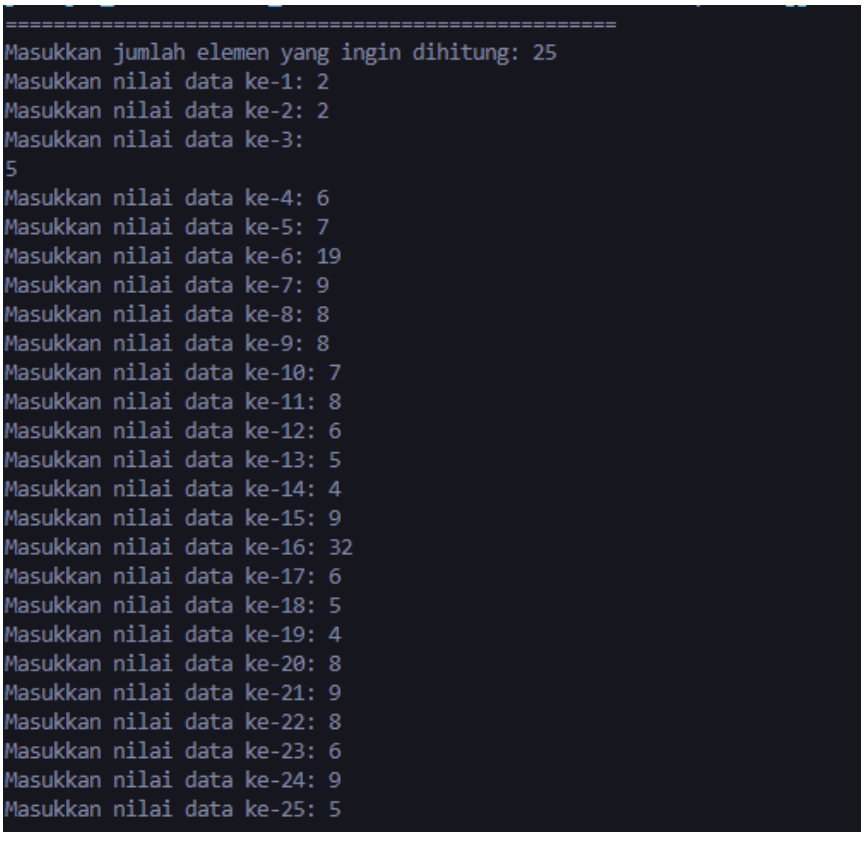
4. Berikut kode program yang sudah ditambahkan pengecekan waktu eksekusi didalam 2 method tersebut

```java
package BruteForceDivideConquer.minggu5;

import java.util.Date;
import java.util.Scanner;

public class MainFaktorial27 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("================================================");
        System.out.print("Masukkan jumlah elemen yang ingin dihitung: ");
        int elemen = sc.nextInt();

        Faktorial27[] fk = new Faktorial27[elemen];
        for (int i = 0; i < elemen; i++) {
            fk[i] = new Faktorial27();
            System.out.print("Masukkan nilai data ke-" + (i + 1) + ": ");
            fk[i].nilai = sc.nextInt();
        }

        System.out.println("================================================");
        System.out.println("Hasil Faktorial dengan Brute Force");
        for (int i = 0; i < elemen; i++) {
            Date startTimeBF = new Date();
            int hasilBF = fk[i].faktorialBF(fk[i].nilai);
            Date endTimeBF = new Date();
            long waktuEksekusiBF = endTimeBF.getTime() - startTimeBF.getTime();
            System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + hasilBF + " (Waktu eksekusi: " + waktuEksekusiBF + " ms)");
        }

        System.out.println("================================================");
        System.out.println("Hasil Faktorial dengan Divide and Conquer");
        for (int i = 0; i < elemen; i++) {
            Date startTimeDC = new Date();
            int hasilDC = fk[i].faktorialDC(fk[i].nilai);
            Date endTimeDC = new Date();
            long waktuEksekusiDC = endTimeDC.getTime() - startTimeDC.getTime();
            System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + hasilDC + " (Waktu eksekusi: " + waktuEksekusiDC + " ms)");
        }
    }
}
```

Kemudian berikut merupakan hasil program nya:



5. Setelah memberikan inputan elemen yang di atas 20 angka maka hasil yang di dapatkan sebagai berikut:

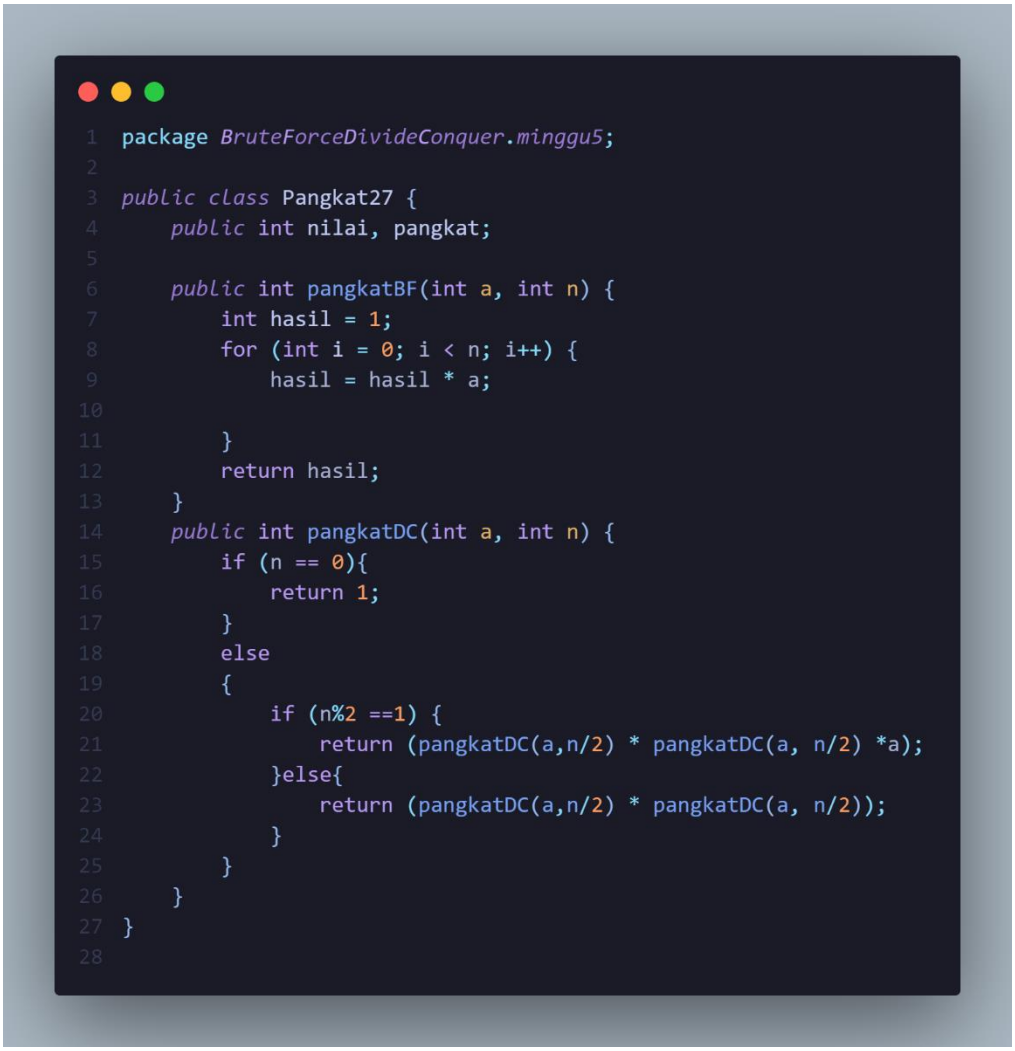


```
================================================
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 2 adalah: 2 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 2 adalah: 2 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 7 adalah: 5040 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 19 adalah: 109641728 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 7 adalah: 5040 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 4 adalah: 24 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 32 adalah: -2147483648 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 4 adalah: 24 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
================================================
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 7 adalah: 5040 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 4 adalah: 24 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 32 adalah: -2147483648 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 4 adalah: 24 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 8 adalah: 40320 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 6 adalah: 720 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 9 adalah: 362880 (Waktu eksekusi: 0 ms)
Faktorial dari nilai 5 adalah: 120 (Waktu eksekusi: 0 ms)
PS D:\COLLEGE\SEMESTER 2\P.STRUKTUR DATA\Jobsheet 4> inputan elemen
di atas 20 angkainputan elemen yang di atas 20 angka
```

4.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

4.3.1 Langkah-langkah Percobaan

Di dalam paket minggu5, buatlah class baru dengan nama Pangkat. Dan di dalam class Pangkat tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya. Berikut merupakan class Pangkat:



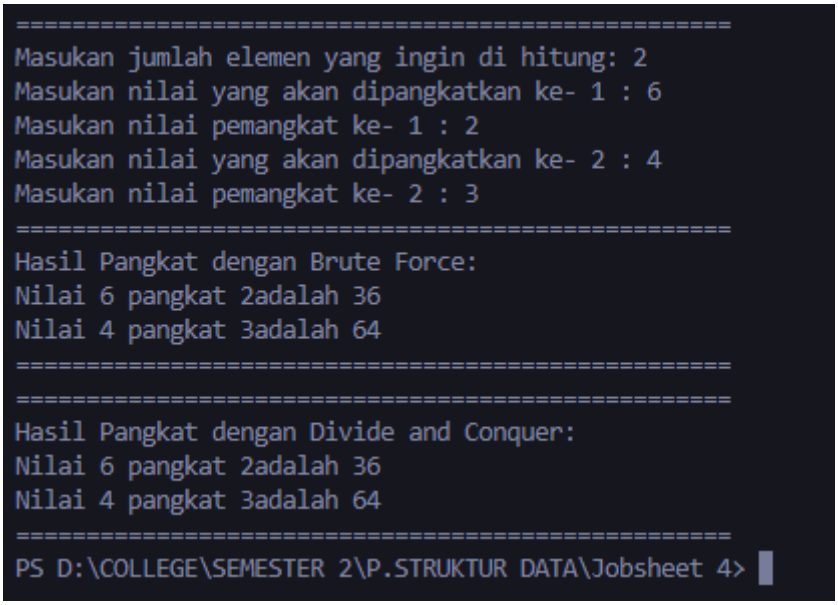
```java
package BruteForceDivideConquer.minggu5;

public class Pangkat27 {
    public int nilai, pangkat;

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

        }
        return hasil;
    }
    public int pangkatDC(int a, int n) {
        if (n == 0){
            return 1;
        }
        else
        {
            if (n%2 ==1) {
                return (pangkatDC(a,n/2) * pangkatDC(a, n/2) *a);
            }else{
                return (pangkatDC(a,n/2) * pangkatDC(a, n/2));
            }
        }
    }
}

```

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

```java
package BruteForceDivideConquer.minggu5;

import java.util.Scanner;

public class MainPangkat27 {
    public static void main(String[] args) {

        Scanner sc27 = new Scanner(System.in);
        System.out.println("================================================");
        System.out.print("Masukan jumlah elemen yang ingin di hitung: ");
        int elemen = sc27.nextInt();

        Pangkat27[] png = new Pangkat27[elemen];

        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukan nilai yang akan dipangkatkan ke- " + (i+1)+ " : ");
            png[i] = new Pangkat27();
            png[i].nilai = sc27.nextInt();
            System.out.print("Masukan nilai pemangkat ke- " + (i+1) + " : ");
            png[i].pangkat = sc27.nextInt();

        }
        System.out.println("================================================");
        System.out.println("Hasil Pangkat dengan Brute Force: ");

        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai " + png[i].nilai + " pangkat " + png[i].pangkat + "adalah " + png[i].pangkatBF(png[i].nilai,png[i].pangkat));
        }
        System.out.println("================================================");


        System.out.println("================================================");
        System.out.println("Hasil Pangkat dengan Divide and Conquer: ");

        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai " + png[i].nilai + " pangkat " + png[i].pangkat + "adalah " + png[i].pangkatDC(png[i].nilai,png[i].pangkat));
        }
        System.out.println("================================================");
    }

}
```

4.3.2 Verifikasi Hasil Percobaan Cocokkan hasil

compile kode program anda dengan gambar berikut ini.



4.3.3 Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu PangkatBF() dan PangkatDC()!

2. Pada method PangkatDC() terdapat potongan program sebagai berikut:

```
if(n%2==1)//bilangan ganjil
    return (pangkatDC(a,n/2)*pangkatDC(a,n/2)*a);
else//bilangan genap
    return (pangkatDC(a,n/2)*pangkatDC(a,n/2));
```

Jelaskan arti potongan kode tersebut!

3. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!
4. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.
5. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!

Jawaban!

1. Perbedaan antara dua metode, yaitu pangkatBF() (Brute Force) dan pangkatDC() (Divide and Conquer), terletak pada pendekatan yang digunakan untuk menghitung hasil pangkat.

   Pangkat Brute Force (pangkatBF()):

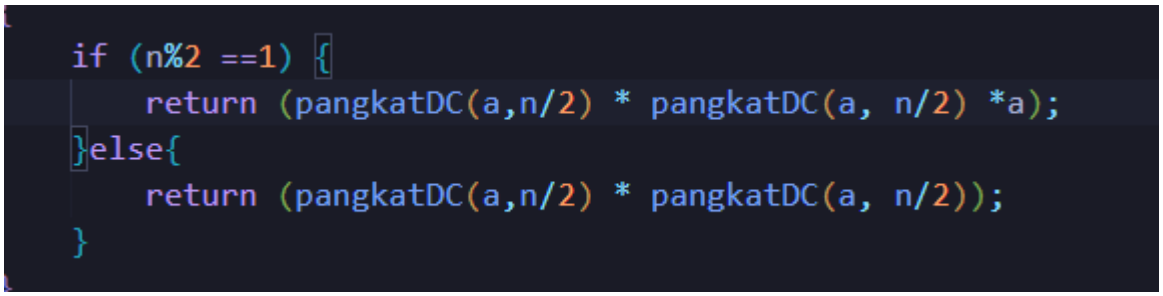
   - Menggunakan pendekatan **Brute Force**.
   - Metode ini menggunakan pendekatan secara langsung, yaitu dengan mengalikan basis (a) sebanyak n kali.
   - Dalam implementasinya, metode ini menggunakan loop for untuk mengalikan basis dengan eksponen sebanyak n kali.
   - Pendekatan ini sederhana dan mudah dipahami, tetapi memiliki kompleksitas waktu O(n), di mana n adalah eksponen.

   Pangkat Divide and Conquer (pangkatDC()):

   - Menggunakan pendekatan **Divide and Conquer**.
   - Metode ini menggunakan pendekatan rekursif untuk memecah masalah menjadi submasalah yang lebih kecil.
   - Ketika eksponen n adalah bilangan ganjil, metode ini memecah masalah menjadi mengalikan pangkat setengah eksponen dengan dirinya sendiri dan mengalikan hasilnya dengan basis.
   - Ketika eksponen n adalah bilangan genap, metode ini memecah masalah menjadi mengalikan pangkat setengah eksponen dengan dirinya sendiri.
   - Pendekatan ini memiliki kompleksitas waktu yang lebih baik daripada Brute Force karena setiap langkahnya hanya memerlukan dua operasi pangkat, sehingga kompleksitas waktu secara kasar adalah O(log n).

2. Berikut merupakan penjelasan dari potongan kode diatas:
   Pengecekan if (n % 2 == 1) digunakan untuk memeriksa apakah eksponen (n) merupakan bilangan ganjil. Jika eksponen ganjil, maka kita harus mengalikan hasil pangkat setengah eksponen dengan dirinya sendiri dan kemudian dikalikan dengan basis (a). Ini adalah langkah tambahan yang dibutuhkan untuk mengatasi sisa hasil bagi yang muncul saat eksponen ganjil.
   Jika eksponen (n) merupakan bilangan ganjil, maka metode memanggil dirinya sendiri untuk menghitung pangkat dari setengah eksponen (pangkatDC(a, n / 2)) dan mengalikannya dengan dirinya sendiri. Selanjutnya, hasilnya dikalikan dengan basis (a) untuk memperoleh hasil akhir.
   Jika eksponen (n) merupakan bilangan genap, maka metode memanggil dirinya sendiri untuk menghitung pangkat dari setengah eksponen (pangkatDC(a, n / 2)) dan mengalikannya dengan dirinya sendiri. Namun, karena eksponen genap tidak memerlukan langkah tambahan seperti pada kasus ganjil, hasil akhirnya langsung dikembalikan.

3. Ya, tahap combine sudah termasuk dalam kode tersebut. Tahap combine terjadi pada proses pengembalian nilai hasil perhitungan pangkat dari submasalah yang lebih kecil ke tingkat yang lebih tinggi dalam rekursi. Dalam kode pangkatDC(), tahap combine terjadi saat nilai-nilai hasil pangkat dari submasalah yang lebih kecil digabungkan untuk menghasilkan nilai pangkat dari masalah asli.
   Perhatikan potongan kode berikut dari pangkatDC():



   Pada kedua kondisi (ganjil dan genap), terdapat rekursi yang memanggil metode pangkatDC() lagi untuk menghitung pangkat dari setengah eksponen. Ketika nilai pangkat dari setengah eksponen sudah dihitung, nilai-nilai tersebut kemudian digunakan untuk menggabungkan solusi dari submasalah yang lebih kecil menjadi solusi akhir untuk masalah yang lebih besar.

   Dengan demikian, dalam potongan kode tersebut, tahap combine terjadi saat nilai-nilai hasil pangkat dari submasalah yang lebih kecil digunakan untuk menghitung nilai pangkat dari masalah asli dengan cara yang tepat, sesuai dengan sifat eksponen ganjil dan genap.

4. Berikut merupakan modifikasi program dengan menganggap proses pengisian atribut dilakukan dengan konstruktor.

```java
package BruteForceDivideConquer.minggu5;
class Faktorial27 {

    public int nilai;

    public Faktorial27(int nilai) {
      this.nilai = nilai;
    }

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

    public int faktorialDC() {
      if (nilai == 1) {
        return 1;
      } else {
        int fakto = nilai * faktorialDC();
        return fakto;
      }
    }
  }

//MODIF 3

```

```java
package BruteForceDivideConquer.minggu5;

import java.util.Date;
import java.util.Scanner;

public class MainFaktorial27 {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("================================================");
    System.out.print("Masukkan jumlah elemen yang ingin dihitung: ");
    int elemen = sc.nextInt();

    Faktorial27[] fk = new Faktorial27[elemen];
    for (int i = 0; i < elemen; i++) {
      System.out.print("Masukkan nilai data ke-" + (i + 1) + ": ");
      int nilai = sc.nextInt();
      fk[i] = new Faktorial27(nilai);
    }

    System.out.println("================================================");
    System.out.println("Hasil Faktorial dengan Brute Force");
    for (int i = 0; i < elemen; i++) {
      Date startTimeBF = new Date();
      int hasilBF = fk[i].faktorialBF();
      Date endTimeBF = new Date();
      long waktuEksekusiBF = endTimeBF.getTime() - startTimeBF.getTime();
      System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + hasilBF + " (Waktu eksekusi: " + waktuEksekusiBF + " ms)");
    }

    System.out.println("================================================");
    System.out.println("Hasil Faktorial dengan Divide and Conquer");
    for (int i = 0; i < elemen; i++) {
      Date startTimeDC = new Date();
      int hasilDC = fk[i].faktorialDC();
      Date endTimeDC = new Date();
      long waktuEksekusiDC = endTimeDC.getTime() - startTimeDC.getTime();
      System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + hasilDC + " (Waktu eksekusi: " + waktuEksekusiDC + " ms)");
    }
  }
}
//MODIF 3
```

5. Berikut merupakan kode yang sudah dimodifikasi dengan ditambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!

```java
package BruteForceDivideConquer.minggu5;

import java.util.Date;
import java.util.Scanner;

public class MainFaktorial27 {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("================================================");

    // Menampilkan menu
    System.out.println("Pilih metode perhitungan faktorial:");
    System.out.println("1. Brute Force");
    System.out.println("2. Divide and Conquer");
    System.out.print("Masukkan pilihan Anda (1/2): ");
    int pilihan = sc.nextInt();

    // Membaca jumlah elemen
    System.out.print("Masukkan jumlah elemen yang ingin dihitung: ");
    int elemen = sc.nextInt();

    Faktorial27[] fk = new Faktorial27[elemen];
    for (int i = 0; i < elemen; i++) {
      System.out.print("Masukkan nilai data ke-" + (i + 1) + ": ");
      int nilai = sc.nextInt();
      fk[i] = new Faktorial27(nilai);
    }

    // Menjalankan metode yang dipilih
    if (pilihan == 1) {
      System.out.println("================================================");
      System.out.println("Hasil Faktorial dengan Brute Force");
      for (int i = 0; i < elemen; i++) {
        Date startTimeBF = new Date();
        int hasilBF = fk[i].faktorialBF();
        Date endTimeBF = new Date();
        long waktuEksekusiBF = endTimeBF.getTime() - startTimeBF.getTime();
        System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + hasilBF + " (Waktu eksekusi: " + waktuEksekusiBF + " ms)");
      }
    } else if (pilihan == 2) {
      System.out.println("================================================");
      System.out.println("Hasil Faktorial dengan Divide and Conquer");
      for (int i = 0; i < elemen; i++) {
        Date startTimeDC = new Date();
        int hasilDC = fk[i].faktorialDC();
        Date endTimeDC = new Date();
        long waktuEksekusiDC = endTimeDC.getTime() - startTimeDC.getTime();
        System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah: " + hasilDC + " (Waktu eksekusi: " + waktuEksekusiDC + " ms)");
      }
    } else {
      System.out.println("Pilihan tidak valid!");
    }
  }
}
//MODIF 4
```

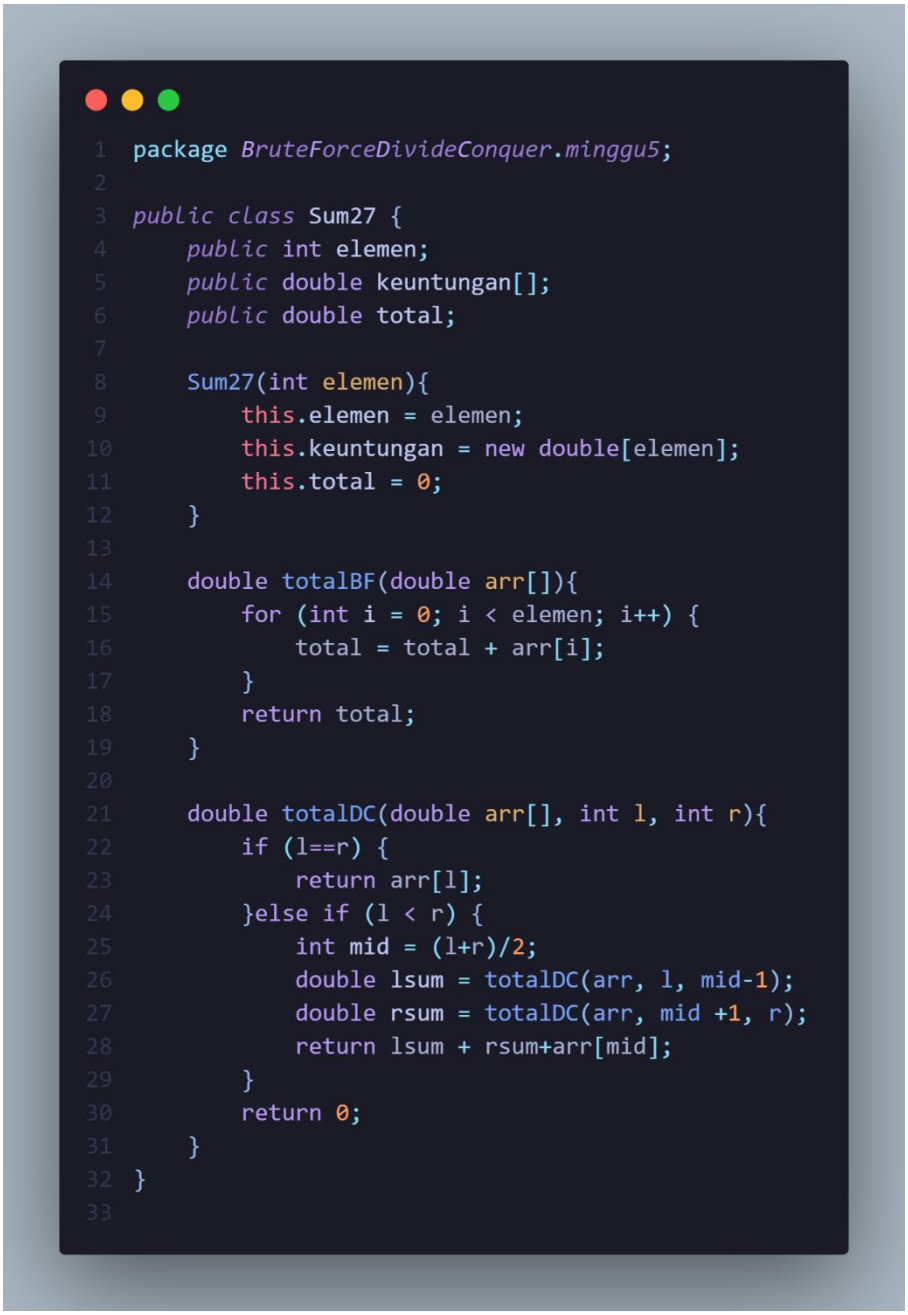
Berikut output program:

```
================================================
Pilih metode perhitungan faktorial:
1. Brute Force
2. Divide and Conquer
Masukkan pilihan Anda (1/2): 1
Masukkan jumlah elemen yang ingin dihitung: 1
Masukkan nilai data ke-1: 1
================================================
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 1 adalah: 1 (Waktu eksekusi: 0 ms)
PS D:\COLLEGE\SEMESTER 2\P.STRUKTUR DATA\Jobsheet 4>
```

4.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

4.4.1 Langkah-langkah Percobaan

Pada paket minggu5. Buat class baru yaitu class Sum. DI salam class tersebut terdapat beberapa atribut jumlah elemen array, array, dan juga total. Tambahkan pula konstruktor pada class Sum. Berikut merupakan class SUM:



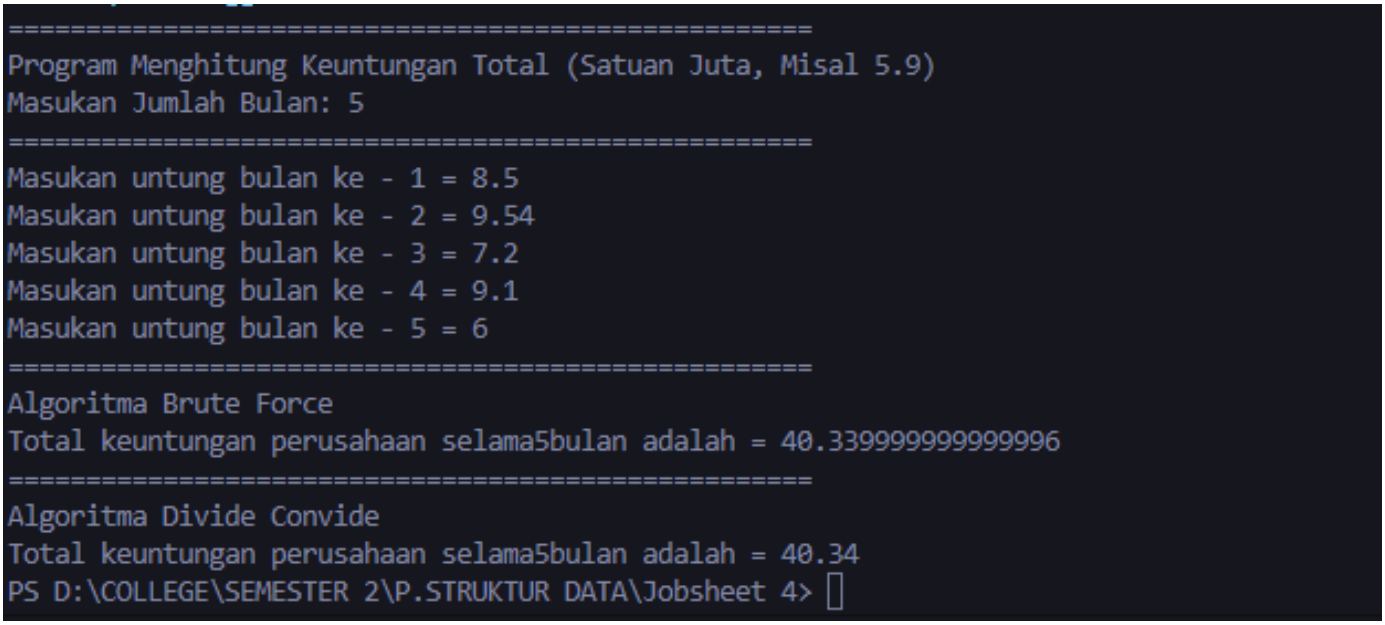
```java
package BruteForceDivideConquer.minggu5;

public class Sum27 {
    public int elemen;
    public double keuntungan[];
    public double total;

    Sum27(int elemen){
        this.elemen = elemen;
        this.keuntungan = new double[elemen];
        this.total = 0;
    }

    double totalBF(double arr[]){
        for (int i = 0; i < elemen; i++) {
            total = total + arr[i];
        }
        return total;
    }

    double totalDC(double arr[], int l, int r){
        if (l==r) {
            return arr[l];
        }else if (l < r) {
            int mid = (l+r)/2;
            double lsum = totalDC(arr, l, mid-1);
            double rsum = totalDC(arr, mid +1, r);
            return lsum + rsum+arr[mid];
        }
        return 0;
    }
}
```

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. Berikut merupakan class MainSum:

```java
package BruteForceDivideConquer.minggu5;

import java.util.Scanner;

public class MainSum27 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("==================================================");
        System.out.println("Program Menghitung Keuntungan Total (Satuan Juta, Misal 5.9)");
        System.out.print("Masukan Jumlah Bulan: ");
        int elm = sc.nextInt();

        Sum27 sm = new Sum27(elm);
        System.out.println("==================================================");
        for (int i = 0; i < sm.elemen; i++) {
            System.out.print("Masukan untung bulan ke - " + (i+1) + " = ");
            sm.keuntungan[i] = sc.nextDouble();
        }

        System.out.println("==================================================");
        System.out.println("Algoritma Brute Force");
        System.out.println("Total keuntungan perusahaan selama" + sm.elemen + "bulan adalah = "+ sm.totalBF(sm.keuntungan));
        System.out.println("==================================================");
        System.out.println("Algoritma Divide Convide");
        System.out.println("Total keuntungan perusahaan selama" + sm.elemen + "bulan adalah = "+ sm.totalDC(sm.keuntungan,0, sm.elemen-1));
    }

}
```

4.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



4.4.3 Pertanyaan

1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method TotalBF() ataupun TotalDC()
2. Perhatikan output dari kedua jenis algoritma tersebut bisa jadi memiliki hasil berbeda di belakang koma. Bagaimana membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma tersebut.
3. Mengapa terdapat formulasi return value berikut?Jelaskan!

```
return lsum+rsum+arr[mid];
```

4. Kenapa dibutuhkan variable mid pada method TotalDC()?
5. Program perhitungan keuntungan suatu perusahaan ini hanya untuk satu perusahaan saja. Bagaimana cara menghitung sekaligus keuntungan beberapa bulan untuk beberapa perusahaan.(Setiap perusahaan bisa saja memiliki jumlah bulan berbeda-beda)? Buktikan dengan program!

Jawaban!

1. Dalam ilustrasi ini, saya akan memberikan contoh perbedaan perhitungan keuntungan menggunakan metode totalBF() (Brute Force) dan totalDC() (Divide and Conquer) dengan data keuntungan perusahaan selama beberapa bulan.

   Misalkan kita memiliki data keuntungan perusahaan selama 6 bulan:

   Metode Brute Force (totalBF()):

   Metode ini akan menambahkan semua keuntungan bulanan secara berurutan dengan menggunakan loop.
   Misalkan data keuntungan perusahaan adalah [5.9, 6.5, 7.2, 8.1, 9.3, 10.2].
   Proses perhitungan dengan metode Brute Force adalah sebagai berikut:

   Total keuntungan = 5.9 + 6.5 + 7.2 + 8.1 + 9.3 + 10.2
   = 47.2
   Jadi, total keuntungan perusahaan selama 6 bulan dengan menggunakan metode Brute Force adalah 47.2 juta.
   Metode Divide and Conquer (totalDC()):

   Metode ini membagi data menjadi dua bagian, menghitung total keuntungan untuk setiap bagian, dan kemudian menggabungkan hasilnya.
   Misalkan data keuntungan perusahaan adalah [5.9, 6.5, 7.2, 8.1, 9.3, 10.2].
   Proses perhitungan dengan metode Divide and Conquer adalah sebagai berikut:

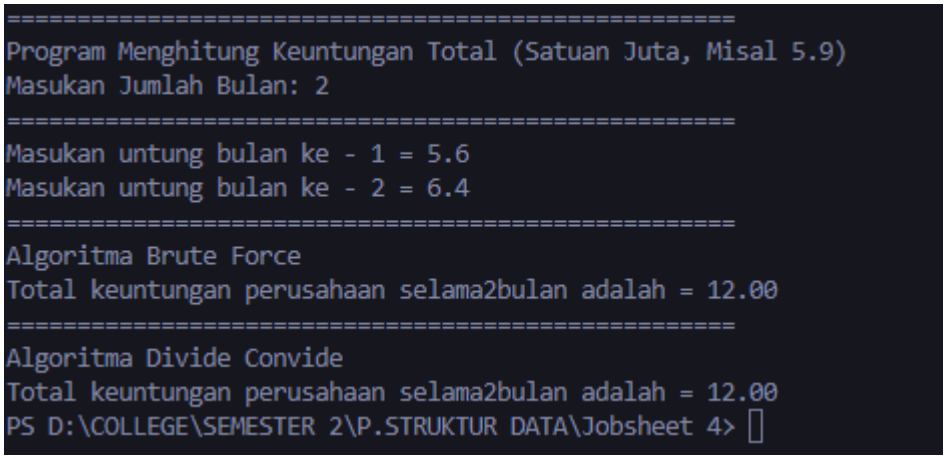
   Bagian kiri: 5.9 + 6.5 + 7.2 = 19.6
   Bagian kanan: 8.1 + 9.3 + 10.2 = 27.6
   Total keuntungan = Bagian kiri + Bagian kanan + nilai tengah
   = 19.6 + 27.6 + 8.1
   = 55.3
   Jadi, total keuntungan perusahaan selama 6 bulan dengan menggunakan metode Divide and Conquer adalah 55.3 juta.
   Dengan demikian, kita dapat melihat perbedaan dalam perhitungan total keuntungan

perusahaan antara metode Brute Force dan metode Divide and Conquer. Metode Divide and Conquer membagi masalah menjadi submasalah yang lebih kecil, menghitung total untuk setiap submasalah, dan kemudian menggabungkan hasilnya untuk mendapatkan total keuntungan keseluruhan.

2. Untuk membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma tersebut, kita dapat menggunakan metode String.format() untuk memformat hasil dengan jumlah digit di belakang koma yang tetap.
   Berikut adalah contoh implementasi untuk membatasi output di belakang koma agar menjadi standar:

```java
package BruteForceDivideConquer.minggu5;

public class Sum27 {
    public int elemen;
    public double keuntungan[];
    public double total;

    Sum27(int elemen){
        this.elemen = elemen;
        this.keuntungan = new double[elemen];
        this.total = 0;
    }

    String totalBF(double arr[]){
        for (int i = 0; i < elemen; i++) {
            total = total + arr[i];
        }
        // Menggunakan String.format() untuk membatasi output di belakang koma
        return String.format("%.2f", total);
    }

    String totalDC(double arr[], int l, int r){
        if (l==r) {
            return String.format("%.2f", arr[l]);
        }else if (l < r) {
            int mid = (l+r)/2;
            double lsum = Double.parseDouble(totalDC(arr, l, mid-1));
            double rsum = Double.parseDouble(totalDC(arr, mid +1, r));
            // Menggunakan String.format() untuk membatasi output di belakang koma
            return String.format("%.2f", lsum + rsum+arr[mid]);
        }
        return "0.00";
    }
}
```

Berikut untuk output program nya:



3. Formulasi return lsum + rsum + arr[mid]; digunakan dalam method totalDC() pada pendekatan Divide and Conquer untuk menghitung total keuntungan dari seluruh elemen. Pada pendekatan Divide and Conquer, method totalDC() akan membagi masalah menjadi submasalah yang lebih kecil hingga mencapai kasus dasar (base case). Setelah mendapatkan hasil dari submasalah yang lebih kecil, method tersebut akan menggabungkan hasil-hasil tersebut untuk memperoleh solusi dari masalah asli. Pada setiap langkah rekursif, nilai lsum dan rsum adalah total keuntungan dari submasalah yang lebih kecil di sebelah kiri dan kanan (bagian kiri dan kanan dari array). Kita ingin menambahkan total keuntungan dari kedua submasalah ini bersama dengan nilai keuntungan dari elemen tengah (arr[mid]) untuk mendapatkan total keuntungan dari seluruh array. Jadi, return lsum + rsum + arr[mid]; adalah formulasi yang tepat untuk mengembalikan total keuntungan dari seluruh array. Dengan demikian, method totalDC() akan mengembalikan total keuntungan dari submasalah yang lebih kecil di bagian kiri, submasalah yang lebih kecil di bagian kanan, dan keuntungan dari elemen tengah, yang digabungkan menjadi total keuntungan dari seluruh array.

4. Variabel mid diperlukan dalam metode totalDC() karena metode ini membagi array menjadi dua bagian yang seimbang secara rekursif. Metode totalDC() membagi array ke dalam dua bagian yang seimbang pada setiap langkah rekursif. Variabel mid digunakan untuk menentukan titik tengah dari array, yang akan menjadi batas antara bagian kiri dan bagian kanan setiap kali array dibagi. Variabel mid digunakan untuk membagi array menjadi dua bagian yang seimbang. Dengan mengetahui indeks tengah, kita dapat memanggil rekursif totalDC() untuk kedua bagian array tersebut dengan mengatur batas l (indeks awal) dan r (indeks akhir) secara sesuai.

5. Untuk menghitung keuntungan beberapa bulan untuk beberapa perusahaan, kita dapat menggunakan pendekatan yang mirip dengan solusi sebelumnya, tetapi kali ini kita akan menggunakan kelas yang merepresentasikan perusahaan dan menyimpan data keuntungan bulanan untuk setiap perusahaan tersebut.

   Berikut adalah contoh implementasi program untuk menghitung keuntungan beberapa bulan untuk beberapa perusahaan:

```java
package BruteForceDivideConquer.minggu5;

import java.text.DecimalFormat;


class Perusahaan {
    private String namaPerusahaan;
    private double[] keuntunganBulanan;

    public Perusahaan(String namaPerusahaan, int jumlahBulan) {
        this.namaPerusahaan = namaPerusahaan;
        this.keuntunganBulanan = new double[jumlahBulan];
    }

    public void setKeuntunganBulanan(int bulan, double keuntungan) {
        this.keuntunganBulanan[bulan - 1] = keuntungan;
    }

    public double hitungTotalKeuntungan() {
        double total = 0;
        for (double keuntungan : keuntunganBulanan) {
            total += keuntungan;
        }
        return total;
    }

    public String getTotalKeuntunganFormatted() {
        DecimalFormat df = new DecimalFormat("#.##");
        return df.format(hitungTotalKeuntungan());
    }

    public String getNamaPerusahaan() {
        return namaPerusahaan;
    }
}
```

```java
package BruteForceDivideConquer.minggu5;

import java.util.Scanner;

public class MainSum27_1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Scanner sc27 = new Scanner(System.in);
        System.out.print("Masukkan jumlah perusahaan: ");
        int jumlahPerusahaan = sc.nextInt();
        sc.nextLine(); // Membuang karakter newline

        Perusahaan[] perusahaan = new Perusahaan[jumlahPerusahaan];

        // Input data keuntungan bulanan untuk setiap perusahaan
        for (int i = 0; i < jumlahPerusahaan; i++) {
            System.out.print("Masukkan nama perusahaan ke-" + (i + 1) + ": ");
            sc.next(); // Membuang karakter newline

            String namaPerusahaan = sc.nextLine();

            System.out.print("Masukkan jumlah bulan untuk perusahaan " + namaPerusahaan + ": ");
            int jumlahBulan = sc27.nextInt();

            perusahaan[i] = new Perusahaan(namaPerusahaan, jumlahBulan);

            System.out.println("Masukkan keuntungan bulanan untuk perusahaan " + namaPerusahaan + ":");
            for (int j = 0; j < jumlahBulan; j++) {
                System.out.print("  Bulan ke-" + (j + 1) + ": ");
                double keuntungan = sc.nextDouble();
                perusahaan[i].setKeuntunganBulanan(j + 1, keuntungan);
            }
        }

        // Menampilkan total keuntungan untuk setiap perusahaan
        for (Perusahaan p : perusahaan) {
            System.out.println(
                    "Total keuntungan " + p.getTotalKeuntunganFormatted() + " juta untuk " + p.getNamaPerusahaan());
        }

        sc.close();
    }
}
```

Berikut hasil output program nya:

```
Masukkan jumlah perusahaan: 2
Masukkan nama perusahaan ke-1: perusahaan1
Masukkan jumlah bulan untuk perusahaan : 3
Masukkan keuntungan bulanan untuk perusahaan :
  Bulan ke-1: 5.6
  Bulan ke-2: 7.9
  Bulan ke-3: 8.3
Masukkan nama perusahaan ke-2: perusahaan2
Masukkan jumlah bulan untuk perusahaan : 4
Masukkan keuntungan bulanan untuk perusahaan :
  Bulan ke-1: 6.8
  Bulan ke-2: 4.5
  Bulan ke-3: 3.2
  Bulan ke-4: 1.6
Total keuntungan 21.8 juta untuk
Total keuntungan 16.1 juta untuk
PS D:\COLLEGE\SEMESTER 2\P.STRUKTUR DATA\Jobsheet 4>
```

4.5 Latihan Praktikum

Buatlah kode program untuk menghitung nilai akar dari suatu bilangan dengan algoritma Brute Force dan Divide Conquer! Jika bilangan tersebut bukan merupakan kuadrat sempurna, bulatkan angka ke bawah.

```java
package BruteForceDivideConquer.minggu5;


public class Latihan27 {
    // Brute Force
     public  double bruteForceSqrt(double x) {
        if (x == 0 || x == 1) {
            return x;
        }

        double result = 1;
        while (result * result <= x) {
            result++;
        }
        return (int) result - 1; // Bulatkan ke bawah
    }

    // Divide Conquer
     public  double divideConquerSqrt(double x) {
        if (x == 0 || x == 1) {
            return x;
        }
        return divideConquerHelper(x, 0, x);
    }

    public  double divideConquerHelper(double x, double start, double end) {
        double mid = start + (end - start) / 2;
        double midSquare = mid * mid;

        if (midSquare == x || (midSquare < x && (mid + 0.001) * (mid + 0.001) > x)) {
            return (int) mid; // Bulatkan ke bawah
        } else if (midSquare < x) {
            return divideConquerHelper(x, mid, end);
        } else {
            return divideConquerHelper(x, start, mid);
        }
    }


}
```

Class Main

```java
import java.util.Scanner;

import BruteForceDivideConquer.minggu5.Latihan27;

public class LatihanMain27 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("================================");
        System.out.println("        Perhitungan Akar");
        System.out.println("================================");
        System.out.print("Masukkan bilangan: ");
        double number = scanner.nextDouble();
        Latihan27 latihan = new Latihan27();

        // Brute Force
        double bruteResult = latihan.bruteForceSqrt(number);
        System.out.println("================================");
        System.out.println("Nilai akar dari  " + number + " adalah: " + bruteResult + " Dengan menggunakan Brute Force");
        System.out.println("================================");

        // Divide Conquer
        double divideConquerResult = latihan.divideConquerSqrt(number);
        System.out.println("Nilai akar dari " + number + " adalah: " + divideConquerResult + " Dengan menggunakan Divide Conquer");
        System.out.println("================================");

        scanner.close();
    }
}
```

Berikut merupakan output ketika program dijalankan:

```
================================
        Perhitungan Akar
================================
Masukkan bilangan: 512
================================
Nilai akar dari  512.0 adalah: 22.0 Dengan menggunakan Brute Force
================================
Nilai akar dari 512.0 adalah: 22.0 Dengan menggunakan Divide Conquer
================================
PS D:\COLLEGE\SEMESTER 2\P.STRUKTUR DATA\Jobsheet 4>  d:; cd 'd:\COLLEGE\SEMESTER 2
\P.STRUKTUR DATA\Jobsheet 4'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enab
le-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ASUS PC\AppDa
ta\Roaming\Code\User\workspaceStorage\ce8854f1ae90806db860190ea3ba6573\redhat.java\
jdt_ws\Jobsheet 4_64662c57\bin' 'BruteForceDivideConquer.minggu5.Latihan27'
================================
        Perhitungan Akar
================================
Masukkan bilangan: 225
================================
Nilai akar dari  225.0 adalah: 15.0 Dengan menggunakan Brute Force
================================
Nilai akar dari 225.0 adalah: 14.0 Dengan menggunakan Divide Conquer
================================
PS D:\COLLEGE\SEMESTER 2\P.STRUKTUR DATA\Jobsheet 4>
```