LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
PRAKTIKUM 5

Oleh:

RANDA HERU KUSUMA

NIM. 2341760009

SIB-1F / 25

D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Link github: https://github.com/randaheru/Praktikum5.git

1. Class faktorial

```
1   public int nilai;
2
```

Membuat Variabel anggota nilai digunakan untuk menyimpan nilai yang akan dihitung faktorialnya.

```
1  public int faktorialBF(int n) {
2        int fakto = 1;
3        for (int i = 1; i <= n; i++) {
4            fakto *= i;
5        }
6        return fakto;
7    }
```

Metode faktorialBF mengimplementasikan perhitungan faktorial dengan pendekatan Brute Force. Ini menggunakan sebuah loop for untuk mengalikan setiap angka dari 1 hingga n untuk mendapatkan faktorialnya.

```
1  public int faktorialDC(int n) {
2        if (n == 1) {
3            return 1;
4        } else {
5            return n * faktorialDC(n - 1);
6        }
7    }
```

Metode faktorialDC mengimplementasikan perhitungan faktorial dengan pendekatan Divide and Conquer. implementasi rekursif. Faktorial dari n dihitung dengan mengalikan n dengan faktorial dari n-1.

```
1      public static void main(String[] args) {
2          Faktorial25 faktorial = new Faktorial25();
3          int n = 10; // Ganti dengan nilai yang ingin Anda hitung faktorialnya
4
5          int hasilBF = faktorial.faktorialBF(n);
6          System.out.println("Hasil Faktorial BF: " + hasilBF);
7
8          int hasilDC = faktorial.faktorialDC(n);
9          System.out.println("Hasil Faktorial DC: " + hasilDC);
10     }
11 }
12
```

Ini membuat objek Faktorial25 menggunakan konstruktor default. Dan nilai yang akan dihitung faktorialnya ditetapkan ke n.

4.2.3 Pertanyaan

1. Jelaskan mengenai base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial! Dalam base line Algoritma Divide Conquer pada class factorial menggunakan pemilihan if (n==1) return 1; yang artinya jika nilai yang akan dimasukkan nantinya adalah 1 maka hasil yang akan ditampilkan oleh program main nantinya adalah 1 dan berfungsi sebagai batas dari divide conquer dimana perulangan kali akan berakhir saat n sudah samadengan 1.
2. Pada implementasi Algoritma Divide and Conquer Faktorial apakah lengkap terdiri dari 3 tahapan divide, conquer, combine? Jelaskan masing-masing bagiannya pada kode program!

a.Divide : membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Dalam kodingan percobaan 1 faktorial, divide ditujukan oleh adanya pemecahan masalah menjadi 2 upa masalah yang diisyaratkan dengan kondisi pemilihan if-else dimana if berperan sebagai base case dan else sebagai rekursif call.

CODING:

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

b.Conquer: memecahkan (menyelesaikan) masing-masing upamasalah (secara rekursif) Dalam kodingan percobaan 1 faktorial,conquer ditujukan oleh adanya penyelesaian masalah secara reqursif dimana upa masalah diselesaikan masing-masing yang telah diisyaratkan pada codingan dalam else yang memberikan rumus int fakto = n * faktorialDC(n-1); yang berarti nanti setiap upa masalah akan dikalikan sendiri-sendiri.

CODING:

```
int faktorialDC(int n)
```

c.Combine: mengabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula. Dalam kodingan percobaan 1 faktorial, combine ditujukan oleh adanya penarikan hasil keseluruhan berupa return atau pengembalian nilai dari proses rekursif pada tahap conquer yang diisyaratkan pada return fakto; di else dalam method faktorialDC();

CODING:

```
return fakto;
```

3. Apakah memungkinkan perulangan pada method faktorialBF() dirubah selain menggunakan for?Buktikan!

Bisa, selama termasuk looping maka jenis looping apapun bisa digunakan pada method faktorialBF() dan saya sudah mencoba membuktikannya dengan looping jenis while dan program tetap berjalan dengan baik.

4. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!

```
long startTime = System.currentTimeMillis(); // Waktu awal Brute Force
            int hasilBF = fk[i].faktorialBF(fk[i].nilai);
            long endTime = System.currentTimeMillis(); // Waktu akhir Brute Force
            long executionTimeBF = endTime - startTime; // Waktu eksekusi Brute Force
```

5. Buktikan dengan inputan elemen yang di atas 20 angka, apakah ada perbedaan waktu eksekusi?

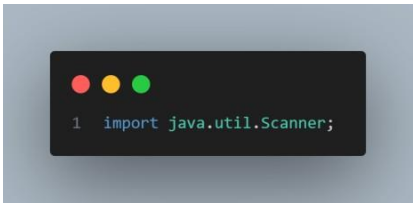
A. Dibawah 20

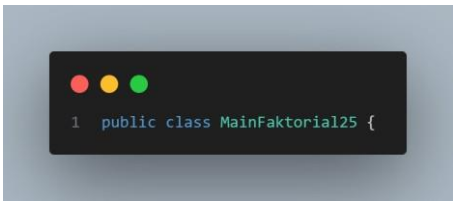
```
==================================================
Masukkan jumlah elemen yang ingin dihitung : 3
Masukkan nilai data ke-1 : 1
Masukkan nilai data ke-2 : 2
Masukkan nilai data ke-3 : 4
==========================================================
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 1 adalah : 1 (Time: 0 milliseconds)
Faktorial dari nilai 2 adalah : 2 (Time: 0 milliseconds)
Faktorial dari nilai 4 adalah : 24 (Time: 0 milliseconds)
==========================================================
Hasil Faktorial dengan Divide and Conquer
Faktorial dari nilai 1 adalah : 1 (Time: 0 milliseconds)
```

B. Diatas 20

```
==================================================
Masukkan jumlah elemen yang ingin dihitung : 21
Masukkan nilai data ke-1 : 1
Masukkan nilai data ke-2 : 2
Masukkan nilai data ke-3 : 3
Masukkan nilai data ke-4 : 4
Masukkan nilai data ke-5 : 5
Masukkan nilai data ke-6 : 6
Masukkan nilai data ke-7 : 7
Masukkan nilai data ke-8 : 8
Masukkan nilai data ke-9 : 9
Masukkan nilai data ke-10 : 2
Masukkan nilai data ke-11 : 3
Masukkan nilai data ke-12 : 4
Masukkan nilai data ke-13 : 5
Masukkan nilai data ke-14 : 6
Masukkan nilai data ke-15 : 7
Masukkan nilai data ke-16 : 8
Masukkan nilai data ke-17 : 9
Masukkan nilai data ke-18 : 3
Masukkan nilai data ke-19 : 4
Masukkan nilai data ke-20 : 5
Masukkan nilai data ke-21 : 6
==========================================================
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 1 adalah : 1 (Time: 0 milliseconds)
Faktorial dari nilai 2 adalah : 2 (Time: 0 milliseconds)
Faktorial dari nilai 3 adalah : 6 (Time: 0 milliseconds)
Faktorial dari nilai 4 adalah : 24 (Time: 0 milliseconds)
Faktorial dari nilai 5 adalah : 120 (Time: 0 milliseconds)
Faktorial dari nilai 6 adalah : 720 (Time: 0 milliseconds)
```

2. Main Faktorial





deklarasi kelas MainFaktorial25

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

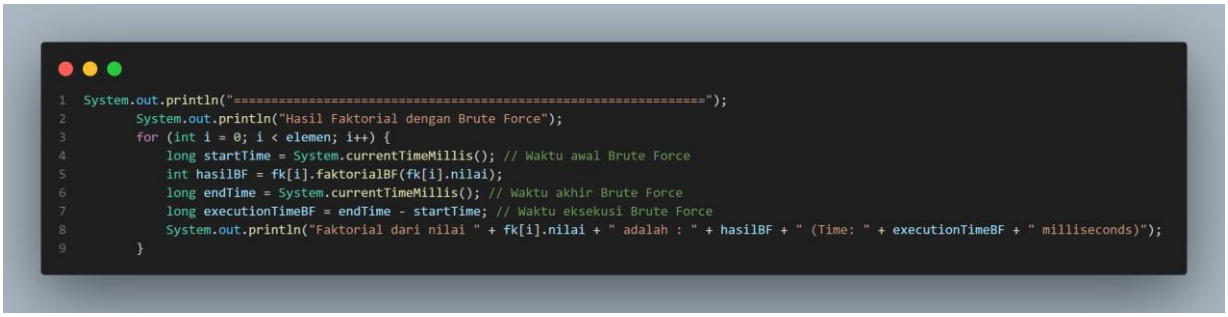
Membuat output untuk meminta jumlah elemen yang ingin dihitung dari pengguna. dan Input disimpan dalam variabel elemen.

```
Faktorial25[] fk = new Faktorial25[elemen];
```

Ini membuat array objek Faktorial25 dengan panjang yang sama dengan jumlah elemen yang dimasukkan oleh pengguna.

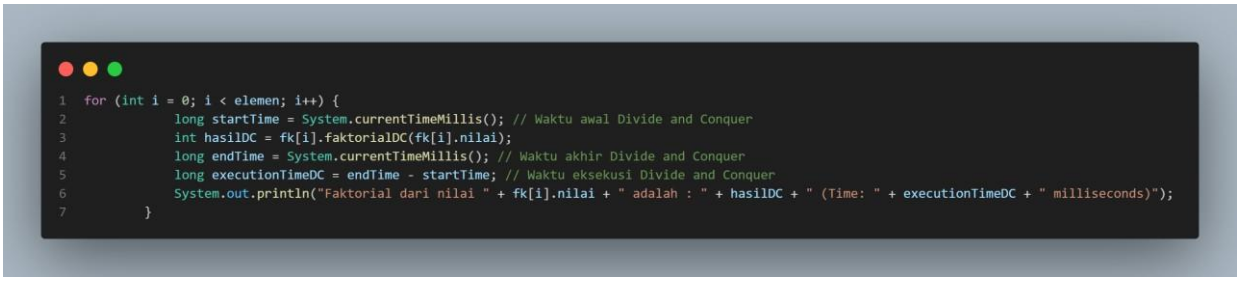
```
for (int i = 0; i < elemen; i++) {
            fk[i] = new Faktorial25();
            System.out.print("Masukkan nilai data ke-" + (i + 1) + " : ");
            fk[i].nilai = sc.nextInt();
```

Ini adalah loop yang meminta input nilai untuk setiap elemen lalu Setiap nilai dimasukkan ke dalam array objek Faktorial25 yang sesuai.



```
System.out.println("============================================================");
        System.out.println("Hasil Faktorial dengan Brute Force");
        for (int i = 0; i < elemen; i++) {
            long startTime = System.currentTimeMillis(); // Waktu awal Brute Force
            int hasilBF = fk[i].faktorialBF(fk[i].nilai);
            long endTime = System.currentTimeMillis(); // Waktu akhir Brute Force
            long executionTimeBF = endTime - startTime; // Waktu eksekusi Brute Force
            System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah : " + hasilBF + " (Time: " + executionTimeBF + " milliseconds)");
        }
```

Menghitung Faktorial dengan Brute Force dan Waktu eksekusi setiap perhitungan dicatat menggunakan System.currentTimeMillis() untuk dihitung.



```
for (int i = 0; i < elemen; i++) {
            long startTime = System.currentTimeMillis(); // Waktu awal Divide and Conquer
            int hasilDC = fk[i].faktorialDC(fk[i].nilai);
            long endTime = System.currentTimeMillis(); // Waktu akhir Divide and Conquer
            long executionTimeDC = endTime - startTime; // Waktu eksekusi Divide and Conquer
            System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah : " + hasilDC + " (Time: " + executionTimeDC + " milliseconds)");
        }
```

Menghitung Faktorial dengan Divide and Conquer

3. Class pangkat

```java
public class Pangkat25 {
    int nilai;
    int pangkat;

```

deklarasi kelas Pangkat25 dengan dua variabel anggota: nilai dan pangkat, yang digunakan untuk menyimpan nilai yang akan dipangkatkan dan pangkatnya.

```java
public Pangkat25(int nilai, int pangkat) {
        this.nilai = nilai;
        this.pangkat = pangkat;
```

Tambahkan konstruktor kelas Pangkat25 yang menerima dua parameter: nilai dan pangkat.

```java
public long pangkatBF(int x, int n) {
        long result = 1;
        for (int i = 0; i < n; i++) {
            result *= x;
        }
        return result;
    }
```

Tambakan metode untuk menghitung pangkat dengan brute force

```java
public long pangkatDC(int x, int n) {
        if (n == 0)
            return 1;
        else if (n % 2 == 0) {
            long temp = pangkatDC(x, n / 2);
            return temp * temp;
        } else {
            long temp = pangkatDC(x, (n - 1) / 2);
            return x * temp * temp;
        }
```

Metode pangkatBF digunakan untuk menghitung pangkat menggunakan pendekatan Brute Force. Ini menggunakan loop for untuk mengalikan nilai x sebanyak n kali.

4. Main pangkat

```
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();

```

Meminta untuk memasukkan jumlah elemen yang ingin dihitung.

```
Pangkat25[] png = new Pangkat25[elemen];

```

Mendeklarasikan sebuah array objek Pangkat25 dengan panjang sesuai dengan jumlah elemen yang dimasukkan pengguna.

```
for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-" + (i + 1) + " : ");
            int nilai = sc.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-" + (i + 1) + " : ");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat25(nilai, pangkat); // Membuat objek Pangkat25 dengan konstruktor
        }
```

Program meminta pengguna untuk memasukkan nilai yang akan dipangkatkan dan nilai pemangkat untuk setiap elemen. Kemudian, objek Pangkat25 dibuat dengan nilai-nilai yang dimasukkan tersebut.

```
System.out.println("============================================================");
        System.out.println("Pilih metode perhitungan pangkat:");
        System.out.println("1. Brute Force");
        System.out.println("2. Divide and Conquer");
        System.out.print("Masukkan pilihan Anda (1 atau 2) : ");
        int pilihan = sc.nextInt();
```

Ditambakan agar bisa meminta pengguna untuk memilih metode perhitungan pangkat: Brute Force atau Divide and Conquer.

```
case 1:
            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));
```

Perhitungan Pangkat dengan Metode Brute Force

```
case 2:
            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));
```

Perhitungan Pangkat dengan Metode Divide and Conquer

4.3.3 Pertanyaan

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

Pada method pangkatBF() operasi mencari hitung hasil pangkat dilakukan dengan cara bruteforce yang dilakukan dengan iterative/perulangan/looping dan algoritma brute forcenya adalah mendeklarasikan dahulu hasil = 1 lalu melakukan perulangan dengan batas n (pangkatnya) dan dalam perulangan tersebut dilakukan looping dari hasil tadi di kali dengan a (bilangan yang akan dipangkat) dan perulangan akan terus berlanjut hingga < n sehingga a akan menghasilkan nilai hasil dari pemangkatannya.

Pada method pangkatDC() operasi mencari hitung hasil pangkat dilakukan dengan cara divide conquer yang dilakukan dengan rekursif dan algoritma divide conquer yang dilakukan terbagi dalam 3 tahap yaitu : divide => memecah masalah jadi upa masalah yang diimplementasikan dalam pemilihan kondisi berupa if-else pada method, lalu ada conquer => penyelesaian dari setiap upa masalah yang tercantum pada else dan terakhir ada combine => menggabungkan kembali menjadi sebuah solusi yang diimplementasikan pada return an di else.

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

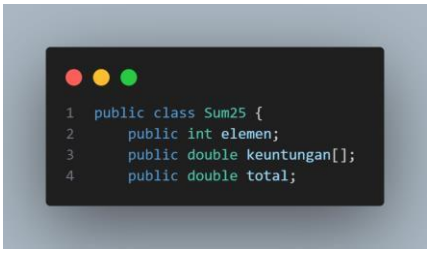
Jika n (pangkat bilangan) dimodulus 2 hasilnya adalah 1 maka returnnya (kembalian nilai) adalah hasil dari (pangkatDC(a,n/2)*pangkatDC(a, n/2)*a) karena bilangan pangkatnya adalah ganjil

Jika n (pangkat bilangan) dimodulus 2 hasilnya adalah tidak sama dengan 1 maka returnnya (kembalian nilai) adalah hasil dari (pangkatDC(a,n/2)*pangkatDC(a, n/2) karena bilangan pangkatnya adalah genap

3. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!

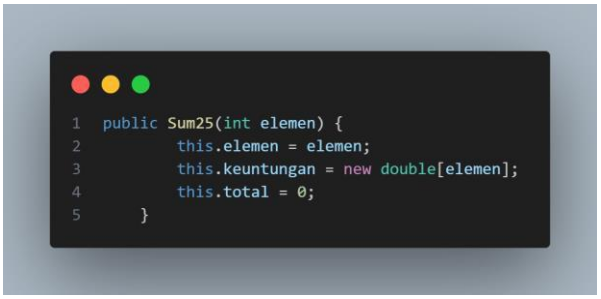
Sudah, tahap combine dalam kode tersebut ditunjukan pada sintaks return atau pengembalian nilai dimana hasil dari conquer atau penyelesaian upa masalah sebelumnya di return kan semua dan dalam tahap combine dilakukan pemanggilan hasil dari bilangan berpangkat tersebut.

5. Class Sum



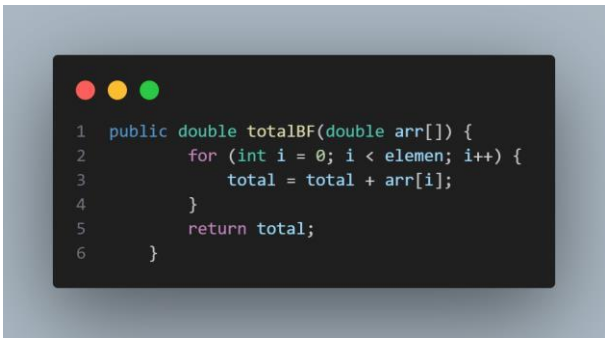
```
public class Sum25 {
    public int elemen;
    public double keuntungan[];
    public double total;
```

Kelas Sum25 dideklarasikan dengan tiga variabel anggota public element, keuntungannya, total



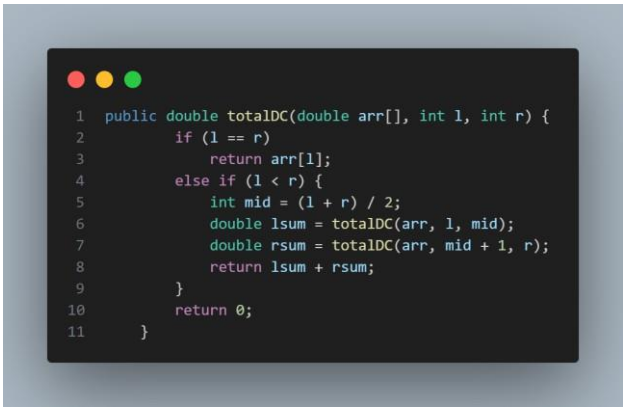
```
public Sum25(int elemen) {
        this.elemen = elemen;
        this.keuntungan = new double[elemen];
        this.total = 0;
    }
```

Lalu menambahkan konstruktor



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

Ditambhakan Metode untuk menghitung total dengan Brute Force



```
public 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);
            double rsum = totalDC(arr, mid + 1, r);
            return lsum + rsum;
        }
        return 0;
    }
```

Tambahkan Metode untuk menghitung total dengan Divide and Conquer

Main Sum

```java
import java.util.Scanner;
```

Mengimpor kelas Scanner dari paket java.util untuk memungkinkan input dari pengguna.

```java
public class MainSum25 {
    public static void main(String[] args) {
```

deklarasi kelas MainSum25 dengan metode main yang merupakan titik masuk utama untuk menjalankan program.

```java
Scanner sc = new Scanner(System.in);
        System.out.println("============================================================");
        System.out.println("Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)");
        System.out.print("Masukkan jumlah perusahaan : ");
        int numCompanies = sc.nextInt();
```

Program meminta pengguna memasukkan jumlah perusahaa. Input tersebut disimpan dalam variabel numCompanies.

```java
Sum25[] companies = new Sum25[numCompanies];
```

Array companies dibuat untuk menyimpan objek Sum25 yang mewakili setiap perusahaan. Ukurannya ditentukan oleh input pengguna.

```java
for (int i = 0; i < numCompanies; i++) {
            System.out.print("Masukkan jumlah bulan untuk perusahaan ke-" + (i + 1) + " : ");
            int numMonths = sc.nextInt();
            companies[i] = new Sum25(numMonths);

            System.out.println("Masukkan keuntungan untuk perusahaan ke-" + (i + 1));
            for (int j = 0; j < companies[i].elemen; j++) {
                System.out.print("Masukkan untung bulan ke-" + (j + 1) + " : ");
                companies[i].keuntungan[j] = sc.nextDouble();
            }
        }
```

Untuk setiap perusahaan, program meminta memasukkan jumlah bulan dan keuntungan untuk masing-masing bulan. Informasi ini disimpan dalam objek Sum25 yang sesuai di dalam array companies.

```
System.out.println("============================================================");
        for (int i = 0; i < numCompanies; i++) {
            System.out.println("Perusahaan ke-" + (i + 1));
            System.out.println("Algoritma Brute Force");
            System.out.println("Total keuntungan perusahaan selama " + companies[i].elemen + " bulan adalah = " + companies[i].totalBF(companies[i].keuntungan));
            System.out.println("Algoritma Divide Conquer");
            System.out.println("Total keuntungan perusahaan selama " + companies[i].elemen + " bulan adalah = " + companies[i].totalDC(companies[i].keuntungan, 0, companies[i].elemen - 1));
            System.out.println("============================================================");
        }
    }
}
```

Tambhkan agar menghitung total keuntungan untuk setiap perusahaan menggunakan metode Brute Force dan Divide and Conquer yang telah didefinisikan di kelas Sum25. Hasilnya dicetak ke layar.

4.4.3 Pertanyaan

1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method TotalBF() ataupun

TotalDC() method totalBF menggunakan perulangan dan di setiap literasi akan menambahkan total dari tiap tiap array key ke var total, sehingga setelah proses perulangan var total berisi total dari value array. Method totalDC memanfaatkan fungsi rekursif, dengan terlebih dahulu memberi validasi_apakah var I sama dengan var r, jika sama maka akan mengembalikan value array key yang sama dengan var l. Kemudian jika tidak maka akan melakukan validasi lagi apakah I < r, jika tidak maka akan mengembalikan 0, jika yam aka akan melakukan kalkulasi rekursif hingga selesai, setelah itu akan mengembalikan basil dari kalkulaşi.

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.

```
System.out.println(x:"======================================================================");
System.out.println(x:"Program Menghitung Keuntungan Total (Satuan Juta. Misal 5,9)");
System.out.print(s:"Masukkan jumlah bulan : ");
int elm = sc.nextInt();

Sum25 sm = new Sum25(elm);
System.out.println(x:"======================================================================");
for(int i=0; i<sm.elemen; i++){
    System.out.print("Masukkan untung bulan ke-"+ (i+1)+" : ");
    sm.keuntungan[i] = sc.nextDouble();
}
```

3. Mengapa terdapat formulasi return value berikut?Jelaskan!

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

untuk menghitung iumlah dari total elemen yang dihitung

4. Kenapa dibutuhkan variable mid pada method TotalDC()?

mid digunakan untuk menampung nilai tengah dan menentukan batas rekursif

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!

```
// Input keuntungan untuk setiap perusahaan
for (int i = 0; i < numCompanies; i++) {
    System.out.print("Masukkan jumlah bulan untuk perusahaan ke-" + (i + 1) + " : ");
    int numMonths = sc.nextInt();
    companies[i] = new Sum25(numMonths);

    System.out.println("Masukkan keuntungan untuk perusahaan ke-" + (i + 1));
    for (int j = 0; j < companies[i].elemen; j++) {
        System.out.print("Masukkan untung bulan ke-" + (j + 1) + " : ");
        companies[i].keuntungan[j] = sc.nextDouble();
    }
```

LANTIHAN

Main Latihan

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

memasukkan sebuah bilangan menggunakan objek Scanner dari kelas java.util.Scanner. Bilangan tersebut disimpan dalam variabel bilangan.

```
int akarBF = Latihan25.akarBF(bilangan);
int akarDC = Latihan25.akarDC(bilangan, low:1, bilangan);
```

Lalu menggunakan metode akarBF dari kelas Latihan25 untuk menghitung akar kuadrat menggunakan pendekatan brute force, dan metode akarDC untuk menghitung akar kuadrat menggunakan pendekatan divide and conquer. Hasilnya disimpan dalam variabel akarBF dan akarDC.

```
System.out.println("Bilangan: " + bilangan);
System.out.println("Akar Kuadrat (Brute Force): " + (akarBF == -1 ? "Bukan kuadrat sempurna" : akarBF));
System.out.println("Akar Kuadrat (Divide & Conquer): " + (akarDC == -1 ? "Bukan kuadrat sempurna" : akarDC));
```

Lalu menampilkan hasil perhitungan akar kuadrat dari kedua metode. Jika hasil adalah -1, artinya bilangan bukanlah sebuah kuadrat sempurna.