Nama : Siti Mutmainah

Kelas : TI-1B

Absen : 21

Mata Kuliah : ALSD

Percobaan 1: Menghitung Nilai Faktorial dengan Algoritm Brute orce dan Divide and Conquer

Langkah-langkah Percobaan

1. Buat folder bernama jobsheet5
2. Buat class bernama Faktorial.java

```
public class Faktorial{
```

3. Lengkapi class dengan method dan atribut

```
public class Faktorial{
```

a) Tambahkan method FaktorialBF()

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

b) Tambahkan method FaktorialDC()

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

c) Tambahkan method main

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

}
```

4. Buat class baru MainFaktorial.java untuk menjalankan class Faktorial

```
public class MainFaktorial {
```

a) Ketik scanner library untuk inputan user

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

b) Sediakan fungsi main sediakan untuuk user memasukkan nilai yang akan dicari faktorialnya

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print(s:"Masukkan nilai:");
    int nilai = input.nextInt();
```

c) Buat objek dari class Faktorialnya dan tampilkan hasil pemanggilan method faktorialBC() dan faktorialDC()

```
Faktorial fk = new Faktorial();
System.out.println("Nilai  Faktorial "+nilai+
"Menggunakan BF: "+fk.faktorialBF(nilai));
System.out.println("Nilai  Faktorial "+nilai+
"Menggunakan DC: "+fk.faktorialDC(nilai));
```

5. Hasil Percobaan

```
Masukkan nilai:5
Nilai  Faktorial 5Menggunakan BF: 120
Nilai  Faktorial 5Menggunakan DC: 120
```

Pertanyaan

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
   Jawab: Jadi if (n==1) itu adalah kondisi dasar (base case), di mana kalau angkanya udah 1, langsung balikin nilai 1 aja artinya perhitungan udah selesai. Sedangkan bagian else itu dipakai kalau angkanya belum 1, jadi bakal terus memanggil fungsi dirinya sendiri (faktorialDC(n-1)) sampai akhirnya ketemu base case tadi
2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
   Jawab: Bisa, menggunakan while

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

```
Masukkan nilai:5
Nilai  Faktorial 5Menggunakan BF: 120
Nilai  Faktorial 5Menggunakan DC: 120
```

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
   Jawab: fakto *= i; artinya nilai fakto dikalikan dengan i terus disimpan lagi ke fakto (biasanya dipakai di perulangan). Sementara int fakto = n * faktorialDC(n-1); itu langsung menghitung dengan cara rekursif, jadi dia kalikan n dengan hasil pemanggilan fungsi faktorial untuk n-1. Yang pertama perulangan, yang kedua rekursif
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
   Jawab: faktorialBF() cara kerjanya pakai perulangan (loop), jadi menghitung faktor satu-satu sampai ke nilai n. Sedangkan faktorialDC() pakai rekursi (memanggil fungsi dirinya sendiri), menghitung dengan cara membagi masalah besar menjadi masalah kecil sampai akhirnya ketemu kondisi dasar. Jadi yang satu iteratif, yang satu rekursif

Percobaan 2: Menghitung Hasil Pangkat dengan Algoritma BF dan DC

Langkah-langkah Percobaan

1. Buat class Pangkat.java lalu buat atribut dan method

```
public class Pangkat {
```

```
int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter

```
Pangkat(int n, int p){
    nilai = n;
    pangkat = p;
}
```

3. Tambahkan method main

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

}
```

4. Tambahkan method PangkatBF()

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

5. Tambahkan method PangkatDC()

```
int pangkatDC(int a, int n){
    if(n==1){
        return a;
    } 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));
        }
```

6. Buat class baru MainPangkat.java, untuk input user jumlah elemen yang dihitung pangkatnya

```
public class MainPangkat {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

    System.out.print(s:"Masukkan jumlah elemen: ");
    int elemen = input.nextInt();
```

7. Buat Scanner library untuk inputan user

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

8. Selanjutnya instansiasi array, tambahkan proses pengisian beberapa nilai yang dipangkatkan sekaligus dengan pemangkatnya

```
Pangkat[] png = new Pangkat[elemen];
for(int i=0;i<elemen;i++){
    System.out.print("Masukkan nilai basis elemen ke-"+(i+1)+": ");
    int basis = input.nextInt();
    System.out.print("Masukkan nilai pangkat elemen ke-"+(i+1)+": ");
    int pangkat = input.nextInt();
    png[i] = new Pangkat(basis, pangkat);
}
```

9. Panggil hasilnya dengan mengeluarkan eturn value dari method PangkatBF() dan PangkatDC()

```
System.out.println(x:"HASIL PANGKAT BRUTEFOCE: ");
for(Pangkat p : png){
    System.out.println(p.nilai+"^"+p.pangkat+": "+p.pangkatBF(p.nilai, p.pangkat));
}  System.out.println(x:"HASIL PANGKAT DIVIDE AND CONQUER: ");
   for(Pangkat p : png){
       System.out.println(p.nilai+"^"+p.pangkat+": "+p.pangkatDC(p.nilai, p.pangkat));
   }
```

10. Hasil Percobaan

```
Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7
HASIL PANGKAT BRUTEFOCE:
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()!
   Jawab: pangkatBF() itu menghitung pangkat dengan cara dihitung satu-satu pakai perulangan (loop). Sedangkan pangkatDC() membagi masalah besar jadi lebih kecil dan menghitungnya secara rekursif
2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!
   Jawab: Iya, combine terjadi saat dia mengalikan hasil pemanggilan rekursif

```
if(n%2==1){
    return (pangkatDC(a, n/2)*pangkatDC(a, n/2)*a);
}else{
    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?
Jawab: Bisa, karena sudah ada atribut nilai dan pangkat di dalam class

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

```
for(Pangkat p : png){
    System.out.println(p.nilai+"^"+p.pangkat+": "+p.pangkatBF());
```

```
Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7
HASIL PANGKAT BRUTEFOCE:
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVIDE AND CONQUER:
2^3: 8
4^5: 1024
6^7: 279936
```

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
   Jawab: Cara kerja pangkatBF() itu menghitung pangkat dengan cara mengalikan angka terus-menerus sebanyak pangkatnya (iteratif). Sedangkan pangkatDC() itu memecah perhitungan menjadi lebih kecil, lalu mengalikan hasil pecahan tersebut (rekursif). Jadi, pangkatBF() itu menghitung satu-satu, sedangkan pangkatDC() menghitung dengan membagi masalah.

Percobaan 3: Menghitung Sum Array dengan Algoritma BF dan DC

Langkah-langkah Percobaan

1. Buat class baru Sum.java

```
public class Sum {
    Run | Debug
```

2. Tambahkan method main

```
public static void main(String[] args) {
}
```

3. Tambahkan konstruktor pada class tersebut

```
double keuntungan[];
Sum(int el){
    keuntungan=new double[el];
}
```

4. Tambahkan method totalBF() yang akan menghitung total nilai array dengan iterative

```
double totalBF(){
    double total=0;
    for(int i=0;i<keuntungan.length;i++){
        total=total+keuntungan[i];
    }
    return total;
}
```

5. Tambahkan method totalCD() utuk implementasi perhitungan nilai total array menggunakan algoritma DC

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

6. Buat class baru MainSum.java

```
public class MainSum {
    Run | Debug
```

7. Tambahkan method main user dapat input berapa bulan keuntungan yang akan dihitung, dan buat instansiasi objek untuk memanggil atribut attau fungsi dalam class Sum

```
public static void main(String[] args) {
    Scanner input = new  Scanner(System.in);
    System.out.print(s:"Masukkan jumlah elemen: ");
    int elemen = input.nextInt();
```

8. Tambajkan Scanner library untuk inputan user

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

9. Buat objek dari class Sum, lakukan perulangan untuk mengambil input nilai keuntungan dan masukkan atribut keuntungan daari objek yang baru dibuat

```
Sum sm = new Sum(elemen);
for(int i=0;i<elemen;i++){
    System.out.print("Masukkan keuntungan ke-"+(i+1)+": ");
    sm.keuntungan[i]=input.nextDouble();
}
```

10. Tampilkan hasil perhitungan melalui objek yang dibuat untuk keduanya

```
System.out.println("Total keuntungan Bruteforce: "+sm.totalBF());
System.out.println("Total Keuntungan menggunakan Divide and Conquer: "+sm.totalDC(sm.keuntungan, l:0, elemen-1));
```

11. Hasil percobaan

```
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 Bruteforce: 150.0
Total Keuntungan menggunakan Divide and Conquer: 150.0
```

Pertanyaan

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

Jawab: Karena mid dipakai untuk membagi array jadi dua bagian. Hitung dari tengah ke kiri sama dari tengah ke kanan, lalu dijumlahkan

2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

```
double lsum = totalDC(arr, l, mid);
double rsum = totalDC(arr, mid+1, r);
```

Jawab: Fungsinya hitung total dari sisi kiri (lsum) dan sisi kanan (rsum). Jadi array-nya dipecah dua, dihitung masing-masing, baru hasilnya dijumlahkan

3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?

```
return lsum+rsum;
```

Jawab: Karena hasil akhir yang dibutuhkan itu total keseluruhan elemen, jadi hasil pecahan kiri (lsum) dan kanan (rsum) harus dijumlahkan, jika tidak dijumlah hasil tidak utuh

4. Apakah base case dari totalDC()?
   Jawab: Base case itu ketika l == r. Artinya udah sampai elemen paling kecil, lalu ngembaliin elemen itu sendiri. Jadi seperti titik berhentinya rekursi.
5. Tarik Kesimpulan tentang cara kerja totalDC()
   Jawab: totalDC() itu kerjanya pakai metode Divide and Coquer akan membagi array jadi dua, hitung bagian kiri dan kanan secara rekursif sampai ketemu elemen terkecil, lalu semua hasilnya dijumlah.