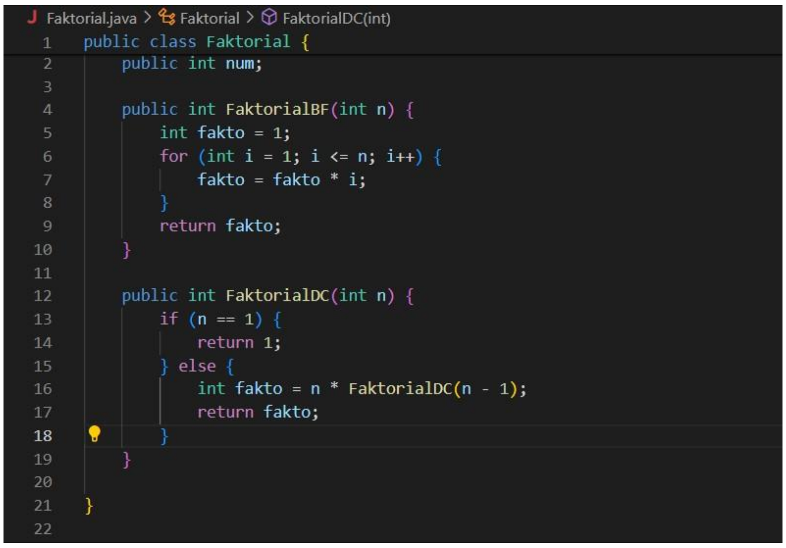
JOBSHEET 5

Algoritma dan Struktur Data

Nama : Yuma Akhunza Kausar Putra

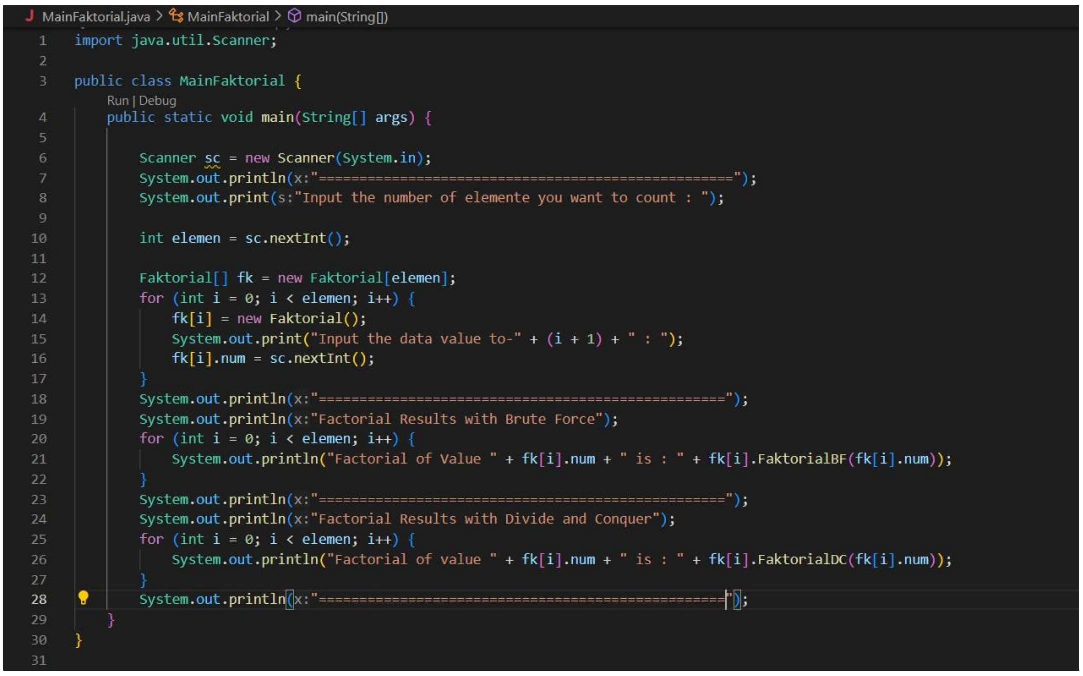
NIM : 2341720259

2.3.1 Practikum



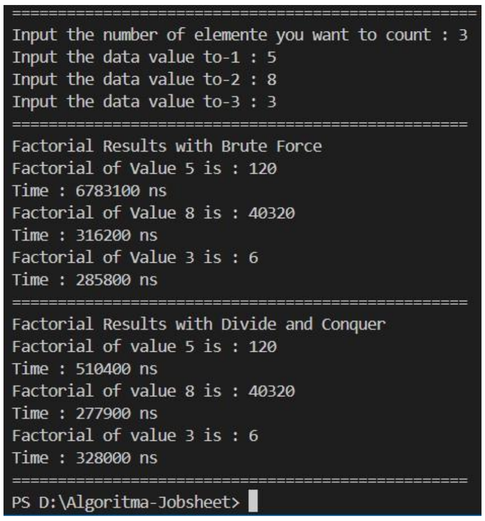
Faktorial.java > Faktorial > FaktorialDC(int)

```java
public class Faktorial {
    public int num;

    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;
        }
    }

}
```



MainFaktorial.java > MainFaktorial > main(String[])

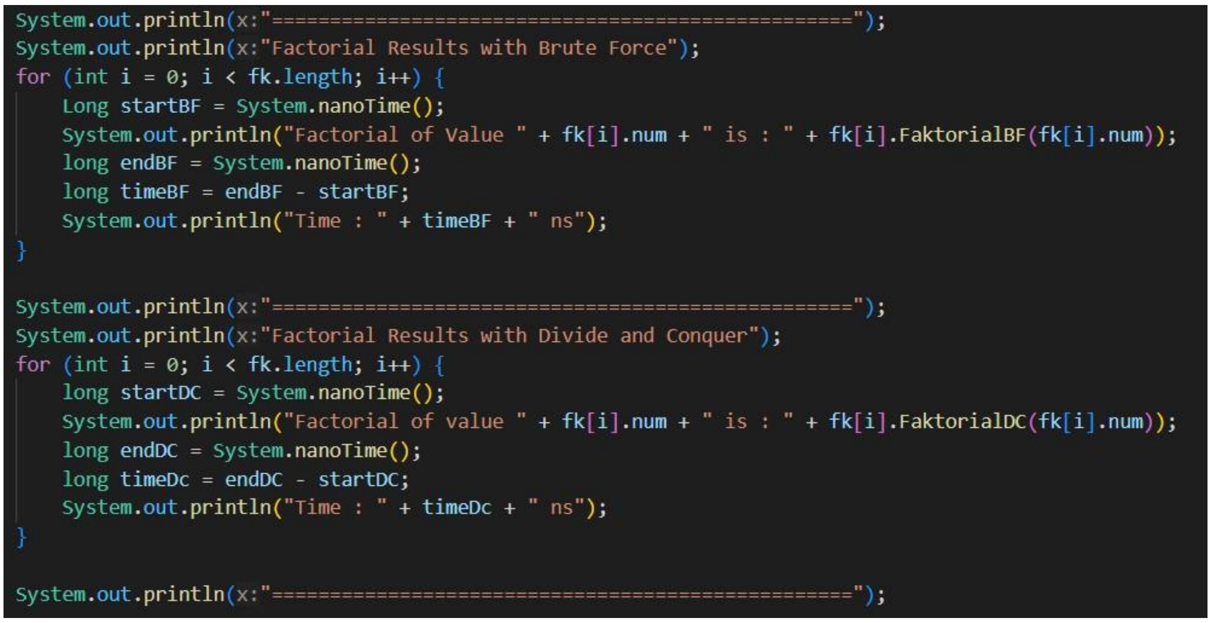
```java
import java.util.Scanner;

public class MainFaktorial {
    Run | Debug
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println(x:"==================================================");
        System.out.print(s:"Input the number of elemente you want to count : ");

        int elemen = sc.nextInt();

        Faktorial[] fk = new Faktorial[elemen];
        for (int i = 0; i < elemen; i++) {
            fk[i] = new Faktorial();
            System.out.print("Input the data value to-" + (i + 1) + " : ");
            fk[i].num = sc.nextInt();
        }
        System.out.println(x:"=================================================");
        System.out.println(x:"Factorial Results with Brute Force");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Factorial of Value " + fk[i].num + " is : " + fk[i].FaktorialBF(fk[i].num));
        }
        System.out.println(x:"=================================================");
        System.out.println(x:"Factorial Results with Divide and Conquer");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Factorial of value " + fk[i].num + " is : " + fk[i].FaktorialDC(fk[i].num));
        }
        System.out.println(x:"=================================================");
    }
}
```



```
=================================================
Input the number of elemente you want to count : 3
Input the data value to-1 : 5
Input the data value to-2 : 8
Input the data value to-3 : 3
=================================================
Factorial Results with Brute Force
Factorial of Value 5 is : 120
Time : 6783100 ns
Factorial of Value 8 is : 40320
Time : 316200 ns
Factorial of Value 3 is : 6
Time : 285800 ns
=================================================
Factorial Results with Divide and Conquer
Factorial of value 5 is : 120
Time : 510400 ns
Factorial of value 8 is : 40320
Time : 277900 ns
Factorial of value 3 is : 6
Time : 328000 ns
=================================================
PS D:\Algoritma-Jobsheet>
```

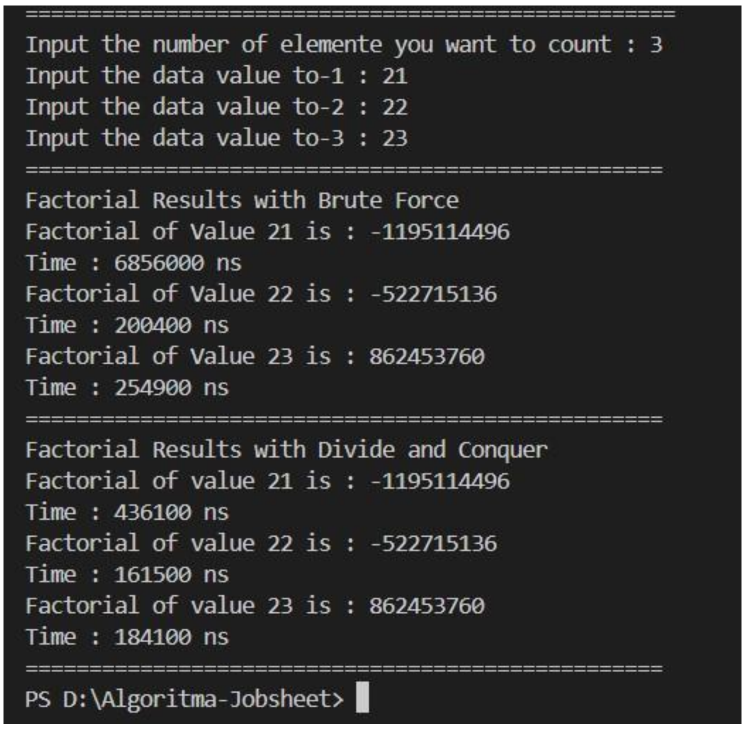
2.3.3 QUESTIONS

1. Explain the Divide Conquer Algorithm for calculating factorial values!
   The Divide and Conquer algorithm for calculating factorial values involves breaking down the problem into smaller subproblems, solving them independently, and then combining the solutions to get the final result.
2. In the implementation of Factorial Divide and Conquer Algorithm is it complete that consists of 3 stages of divide, conquer, combine? Explain each part of the program code!
   - Divide : The division step is implicitly handled by the recursive nature of the algorithm. When calculating the factorial of a number n, the problem is divided into a smaller subproblem of calculating the factorial of n-1.
   - Conquer: The conquer step is where the base case is defined and the recursion stops. In the Factorial Divide and Conquer Algorithm, the base case is when n reaches 1, where the factorial is defined as 1.
   - Combine: In the combine step, the results from the subproblems are combined to get the final result. In this algorithm, the results are combined in a recursive manner as the function calls are resolved and the final result is calculated.
3. Is it possible to repeat the factorial BF () method instead of using for? Prove it! Yes, it is possible to implement the factorial calculation without using a for loop by utilizing a recursive approach which has already been explained in Divide and Conquer Algorithm.
4. Add a check to the execution time of the two types of methods!



```java
System.out.println(x:"=================================================");
System.out.println(x:"Factorial Results with Brute Force");
for (int i = 0; i < fk.length; i++) {
    Long startBF = System.nanoTime();
    System.out.println("Factorial of Value " + fk[i].num + " is : " + fk[i].FaktorialBF(fk[i].num));
    long endBF = System.nanoTime();
    long timeBF = endBF - startBF;
    System.out.println("Time : " + timeBF + " ns");
}

System.out.println(x:"=================================================");
System.out.println(x:"Factorial Results with Divide and Conquer");
for (int i = 0; i < fk.length; i++) {
    long startDC = System.nanoTime();
    System.out.println("Factorial of value " + fk[i].num + " is : " + fk[i].FaktorialDC(fk[i].num));
    long endDC = System.nanoTime();
    long timeDc = endDC - startDC;
    System.out.println("Time : " + timeDc + " ns");
}

System.out.println(x:"=================================================");
```

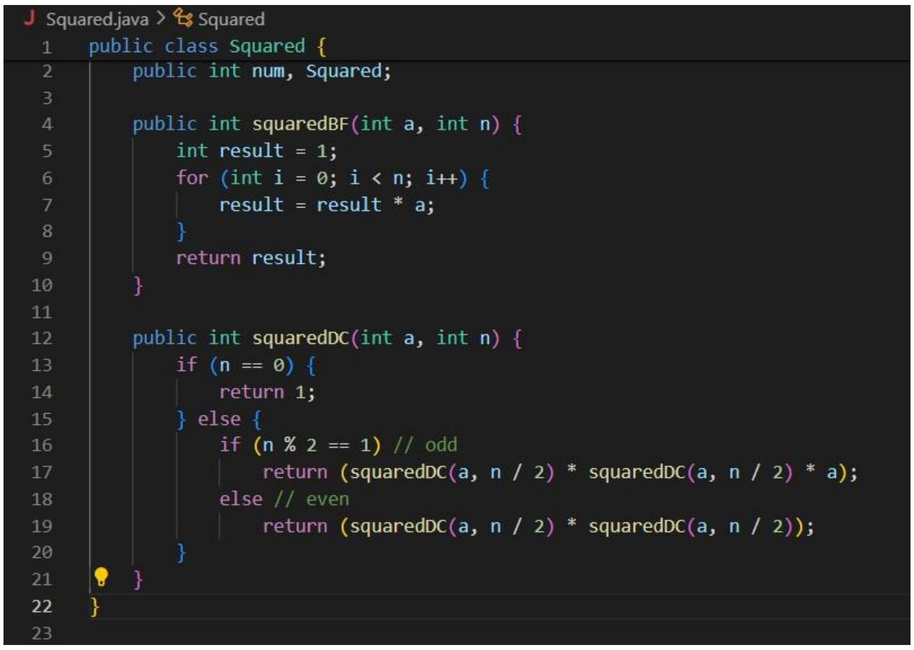
5. Prove by inputting elements that are above 20 digits, is there a difference in execution time?



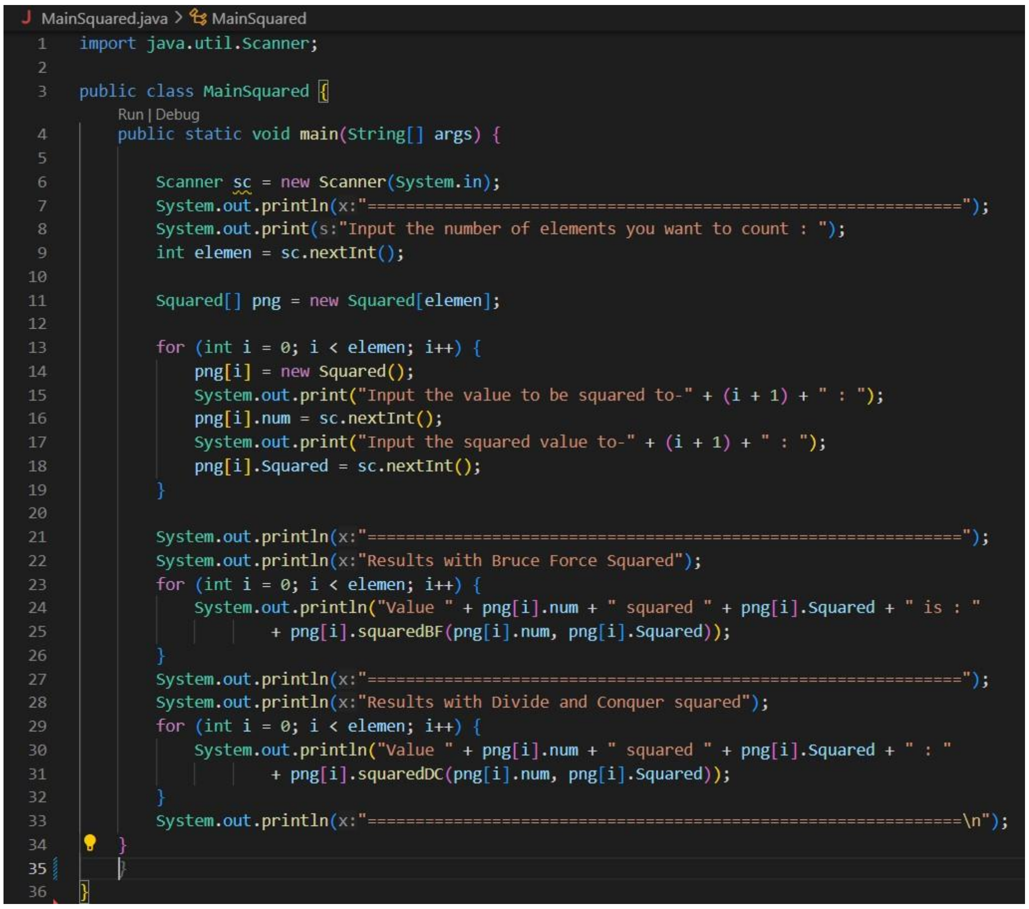
```
=================================================
Input the number of elemente you want to count : 3
Input the data value to-1 : 21
Input the data value to-2 : 22
Input the data value to-3 : 23
=================================================
Factorial Results with Brute Force
Factorial of Value 21 is : -1195114496
Time : 6856000 ns
Factorial of Value 22 is : -522715136
Time : 200400 ns
Factorial of Value 23 is : 862453760
Time : 254900 ns
=================================================
Factorial Results with Divide and Conquer
Factorial of value 21 is : -1195114496
Time : 436100 ns
Factorial of value 22 is : -522715136
Time : 161500 ns
Factorial of value 23 is : 862453760
Time : 184100 ns
=================================================
PS D:\Algoritma-Jobsheet> 
```

2.4 Calculating Squared Results with Brute Force and Divide and Conquer Algorithms

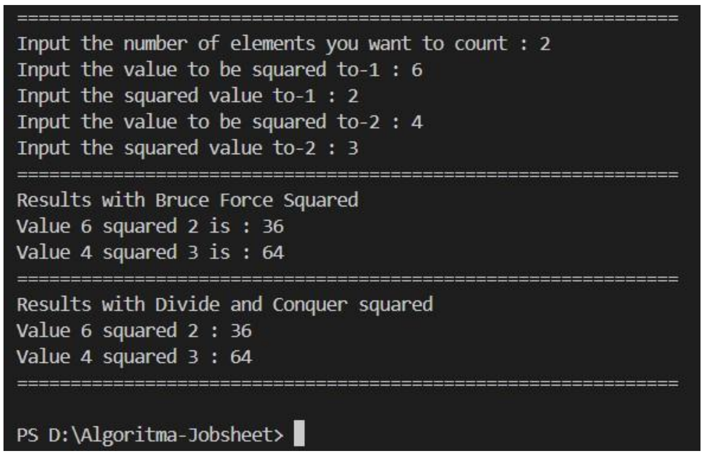
2.4.1 Practicum



```
Squared.java > Squared
public class Squared {
    public int num, Squared;

    public int squaredBF(int a, int n) {
        int result = 1;
        for (int i = 0; i < n; i++) {
            result = result * a;
        }
        return result;
    }

    public int squaredDC(int a, int n) {
        if (n == 0) {
            return 1;
        } else {
            if (n % 2 == 1) // odd
                return (squaredDC(a, n / 2) * squaredDC(a, n / 2) * a);
            else // even
                return (squaredDC(a, n / 2) * squaredDC(a, n / 2));
        }
    }
}
```



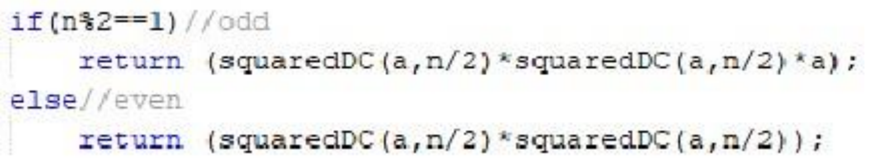
```
MainSquared.java > MainSquared
import java.util.Scanner;

public class MainSquared {
    Run | Debug
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println(x:"==============================================================");
        System.out.print(s:"Input the number of elements you want to count : ");
        int elemen = sc.nextInt();

        Squared[] png = new Squared[elemen];

        for (int i = 0; i < elemen; i++) {
            png[i] = new Squared();
            System.out.print("Input the value to be squared to-" + (i + 1) + " : ");
            png[i].num = sc.nextInt();
            System.out.print("Input the squared value to-" + (i + 1) + " : ");
            png[i].Squared = sc.nextInt();
        }

        System.out.println(x:"==============================================================");
        System.out.println(x:"Results with Bruce Force Squared");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Value " + png[i].num + " squared " + png[i].Squared + " is : "
                    + png[i].squaredBF(png[i].num, png[i].Squared));
        }
        System.out.println(x:"==============================================================");
        System.out.println(x:"Results with Divide and Conquer squared");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Value " + png[i].num + " squared " + png[i].Squared + " : "
                    + png[i].squaredDC(png[i].num, png[i].Squared));
        }
        System.out.println(x:"==============================================================\n");
    }
}
```



```
============================================================
Input the number of elements you want to count : 2
Input the value to be squared to-1 : 6
Input the squared value to-1 : 2
Input the value to be squared to-2 : 4
Input the squared value to-2 : 3
============================================================
Results with Bruce Force Squared
Value 6 squared 2 is : 36
Value 4 squared 3 is : 64
============================================================
Results with Divide and Conquer squared
Value 6 squared 2 : 36
Value 4 squared 3 : 64
============================================================

PS D:\Algoritma-Jobsheet>
```

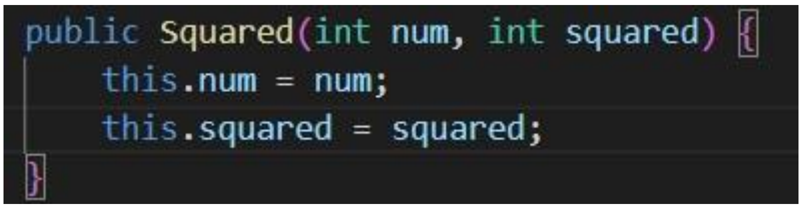
2.4.3 QUESTIONS

1. Explain the differences between the 2 methods made are SquaredBF () an SquaredDC ()!
   The main differences between these two methods are Time Complexity. The brute force approach (SquaredBF()) has a time complexity of O(n), where n is the input number. This is because it multiplies the number a by itself n times. The divide and conquer approach (SquaredDC()) has a time complexity of O(log n), where n is the input number. This is because it recursively divides the number by two until it reaches 0 or 1, and then multiplies the results together.
2. In the SuaredDC () method there is a program as follows:

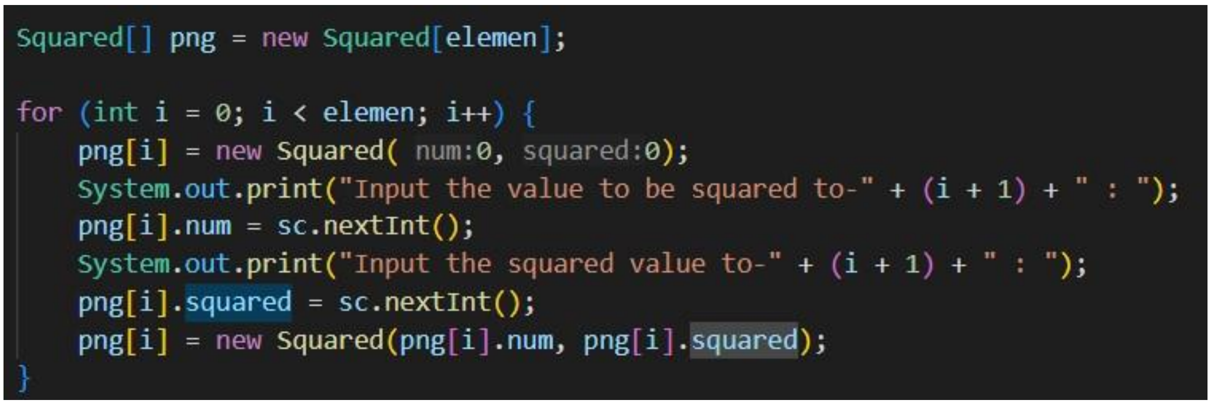
   

   ```
   if(n%2==1)//odd
       return (squaredDC(a,n/2)*squaredDC(a,n/2)*a);
   else//even
       return (squaredDC(a,n/2)*squaredDC(a,n/2));
   ```

   Explain the meaning of the code!
   It checks if the value of n is odd or even. If n is odd, it recursively calls the SquaredDC method with n/2 as the parameter and multiplies the result by itself and a. If n is even, it recursively calls the SquaredDC method with n/2 as the parameter and multiplies the result by itself.
3. Explain whether the combine stage is included in the code!
   In the code, the "combine" stage is on the SquareDC() method, which is implicitly included in the multiplication operations (`*`) performed within the `if` and `else` blocks. After recursively calculating the square of `a` raised to the power of `n/2`, these results are multiplied together. This combines the solutions of the subproblems to obtain the final result.
4. Modification of the program code, assuming the attribute filling process is done by constructor.

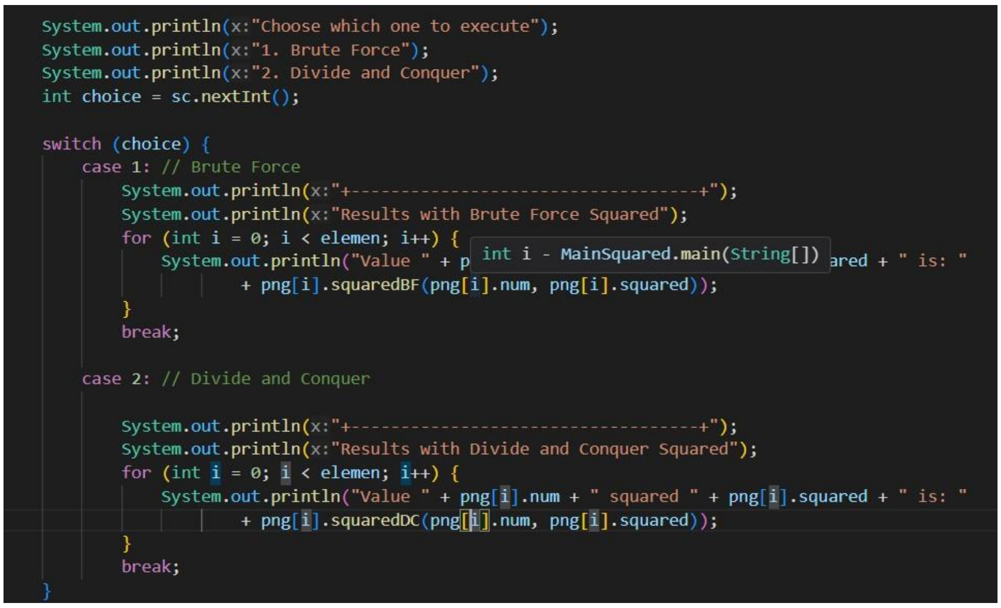


```java
public Squared(int num, int squared) {
    this.num = num;
    this.squared = squared;
}
```

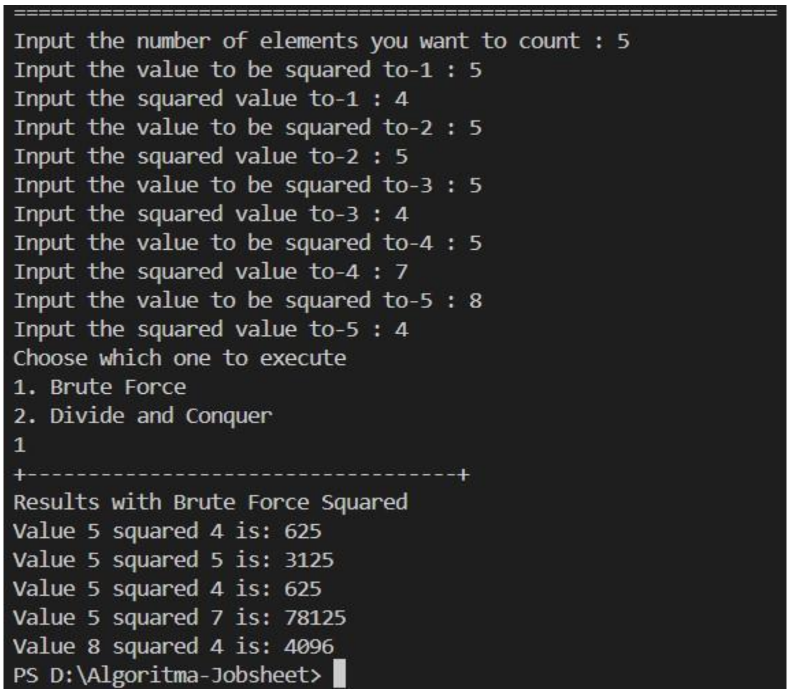


```java
Squared[] png = new Squared[elemen];

for (int i = 0; i < elemen; i++) {
    png[i] = new Squared( num:0, squared:0);
    System.out.print("Input the value to be squared to-" + (i + 1) + " : ");
    png[i].num = sc.nextInt();
    System.out.print("Input the squared value to-" + (i + 1) + " : ");
    png[i].squared = sc.nextInt();
    png[i] = new Squared(png[i].num, png[i].squared);
}
```

5. Add a menu so that only one of the selected methods will be run



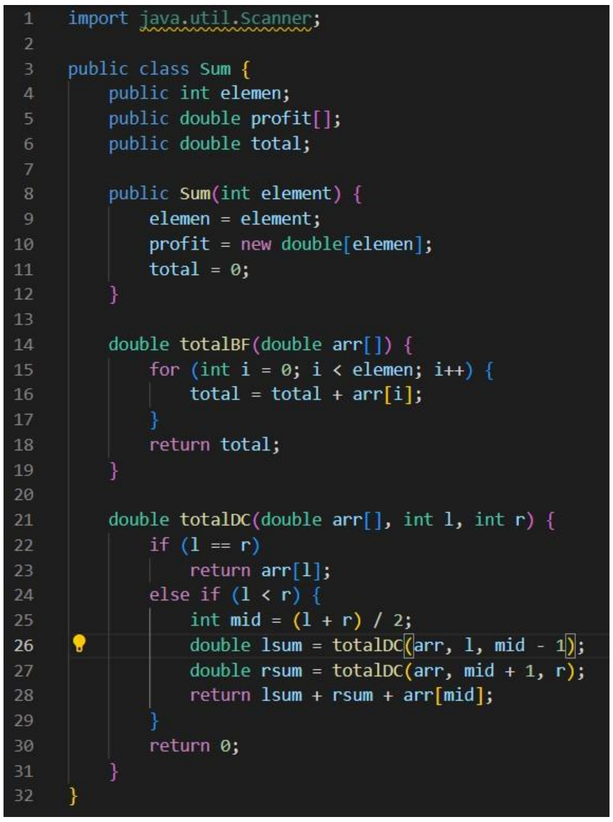
```
System.out.println(x:"Choose which one to execute");
System.out.println(x:"1. Brute Force");
System.out.println(x:"2. Divide and Conquer");
int choice = sc.nextInt();

switch (choice) {
    case 1: // Brute Force
        System.out.println(x:"+----------------------------------+");
        System.out.println(x:"Results with Brute Force Squared");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Value " + p int i - MainSquared.main(String[]) ared + " is: "
                    + png[i].squaredBF(png[i].num, png[i].squared));
        }
        break;

    case 2: // Divide and Conquer

        System.out.println(x:"+----------------------------------+");
        System.out.println(x:"Results with Divide and Conquer Squared");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Value " + png[i].num + " squared " + png[i].squared + " is: "
                    + png[i].squaredDC(png[i].num, png[i].squared));
        }
        break;
}
```



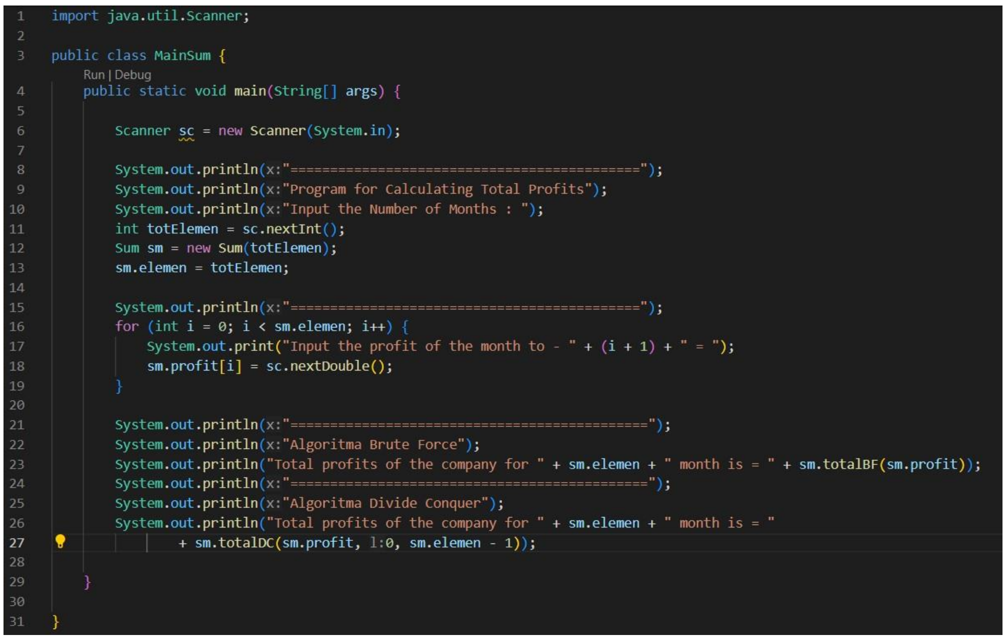
```
==============================================================
Input the number of elements you want to count : 5
Input the value to be squared to-1 : 5
Input the squared value to-1 : 4
Input the value to be squared to-2 : 5
Input the squared value to-2 : 5
Input the value to be squared to-3 : 5
Input the squared value to-3 : 4
Input the value to be squared to-4 : 5
Input the squared value to-4 : 7
Input the value to be squared to-5 : 8
Input the squared value to-5 : 4
Choose which one to execute
1. Brute Force
2. Divide and Conquer
1
+-----------------------------------+
Results with Brute Force Squared
Value 5 squared 4 is: 625
Value 5 squared 5 is: 3125
Value 5 squared 4 is: 625
Value 5 squared 7 is: 78125
Value 8 squared 4 is: 4096
PS D:\Algoritma-Jobsheet>
```

2.5 Calculating Sum Array with Brute Force and Divide and Conquer Algorithms

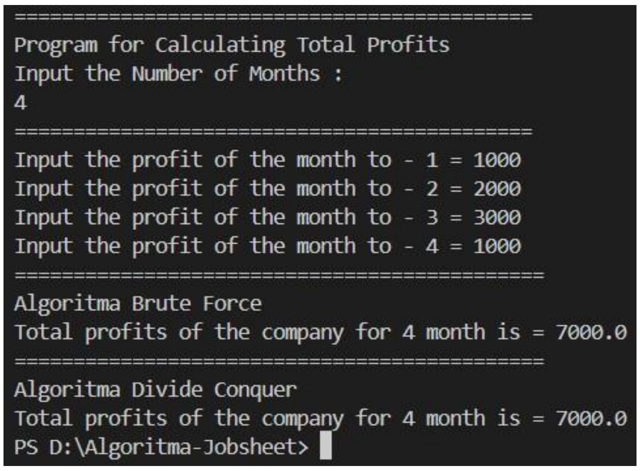
2.5.1 Practicum



```java
import java.util.Scanner;

public class Sum {
    public int elemen;
    public double profit[];
    public double total;

    public Sum(int element) {
        elemen = element;
        profit = new double[elemen];
        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;
    }
}
```



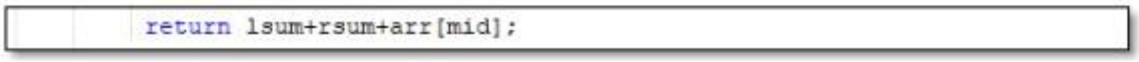
```java
import java.util.Scanner;

public class MainSum {
    Run | Debug
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println(x:"============================================");
        System.out.println(x:"Program for Calculating Total Profits");
        System.out.println(x:"Input the Number of Months : ");
        int totElemen = sc.nextInt();
        Sum sm = new Sum(totElemen);
        sm.elemen = totElemen;

        System.out.println(x:"============================================");
        for (int i = 0; i < sm.elemen; i++) {
            System.out.print("Input the profit of the month to - " + (i + 1) + " = ");
            sm.profit[i] = sc.nextDouble();
        }

        System.out.println(x:"============================================");
        System.out.println(x:"Algoritma Brute Force");
        System.out.println("Total profits of the company for " + sm.elemen + " month is = " + sm.totalBF(sm.profit));
        System.out.println(x:"============================================");
        System.out.println(x:"Algoritma Divide Conquer");
        System.out.println("Total profits of the company for " + sm.elemen + " month is = "
                + sm.totalDC(sm.profit, l:0, sm.elemen - 1));

    }

}
```



```
=============================================
Program for Calculating Total Profits
Input the Number of Months :
4
=============================================
Input the profit of the month to - 1 = 1000
Input the profit of the month to - 2 = 2000
Input the profit of the month to - 3 = 3000
Input the profit of the month to - 4 = 1000
==============================================
Algoritma Brute Force
Total profits of the company for 4 month is = 7000.0
==============================================
Algoritma Divide Conquer
Total profits of the company for 4 month is = 7000.0
PS D:\Algoritma-Jobsheet>
```

2.5.2 QUESTIONS

1. Give an illustration of the difference in profit calculation with the TotalBF () or TotalDC () method.
   Example :
   Suppose we have the following profits for four months: 100.000, 200.000, 300.000, and
   400.000.
   Brute Force Method (`totalBF`):
   - The total profit is calculated by simply summing up all the profits: [100.000 + 200.000 + 300.000 + 400.000 = 1.000.000] Divide and Conquer Method (`totalDC`):
   - The array is divided into two halves: [100.000, 200.000] and [300.000, 400.000]. - The total profit for each half is calculated recursively:
   - For [100.000, 200.000]: 100.000 + 200.000 = 300.000
   - For [300.000, 400.000]: 300.000 + 400.000 = 700.000
   - The results are combined: 300.000 (from the first half) + 700.000 (from the second half) = 1.000.000
2. Why is there the following return value? Explain!

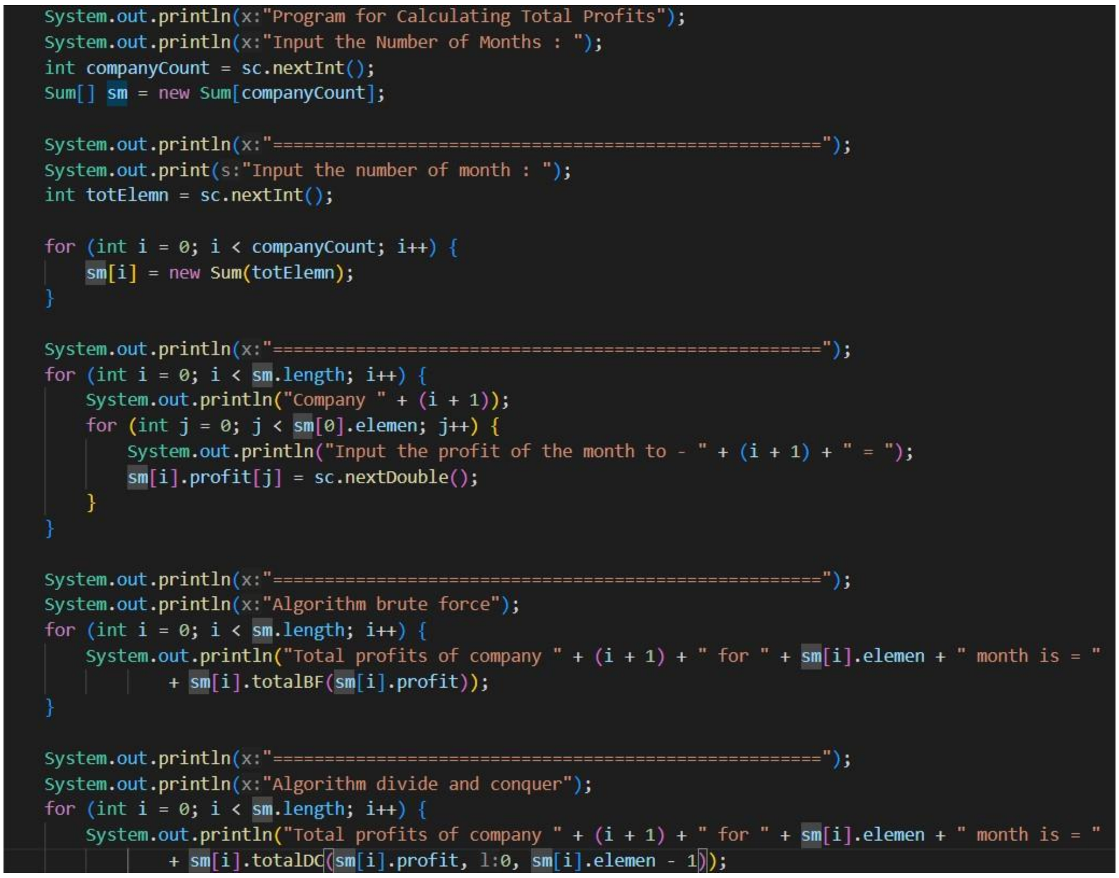


```
return lsum+rsum+arr[mid];
```

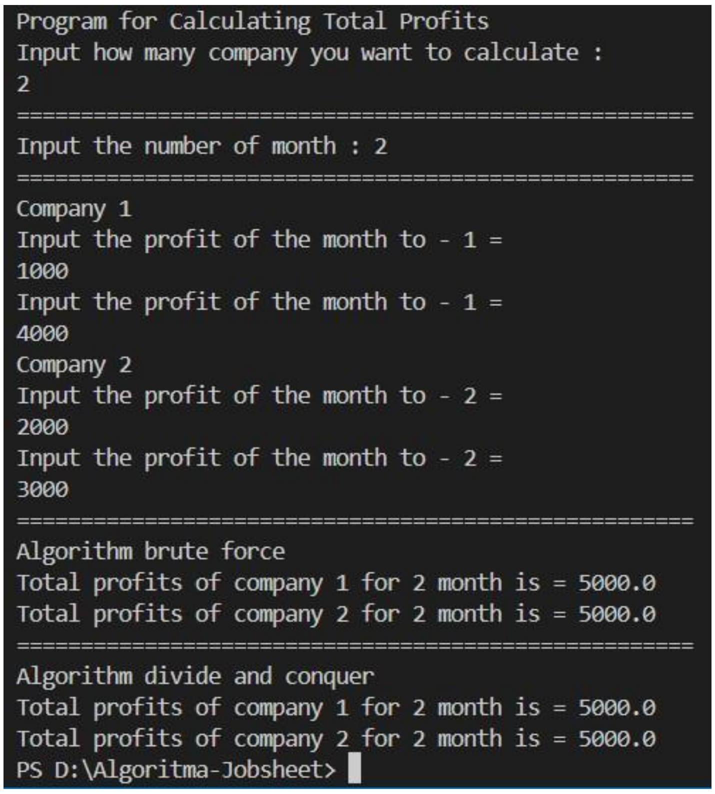
   The return values are crucial for the recursive logic of the divide and conquer approach, allowing the method to combine the results of smaller subproblems to find the total sum of the array.
3. Why is the mid variable required for the TotalDC () method?
   The `mid` variable in the `totalDC` method is required to split the array into two halves recursively. In the divide and conquer approach, the array is divided into smaller subarrays until each subarray contains only one element (base case). The `mid` variable determines the middle index of the current subarray, which helps in dividing the array into two halves: one from index `l` to `mid`, and the other from index `mid + 1` to `r`. Without the `mid` variable, the method wouldn't know where to split the array into two halves, and the divide and conquer approach wouldn't be

possible. Therefore, the `mid` variable plays a crucial role in dividing the array and facilitating the divide and conquer strategy for solving the problem.

4. The profit calculation program for a company is only for one company. How do you calculate several months of profit for several companies at once (each company can have a different number of months)? Prove it with the program!



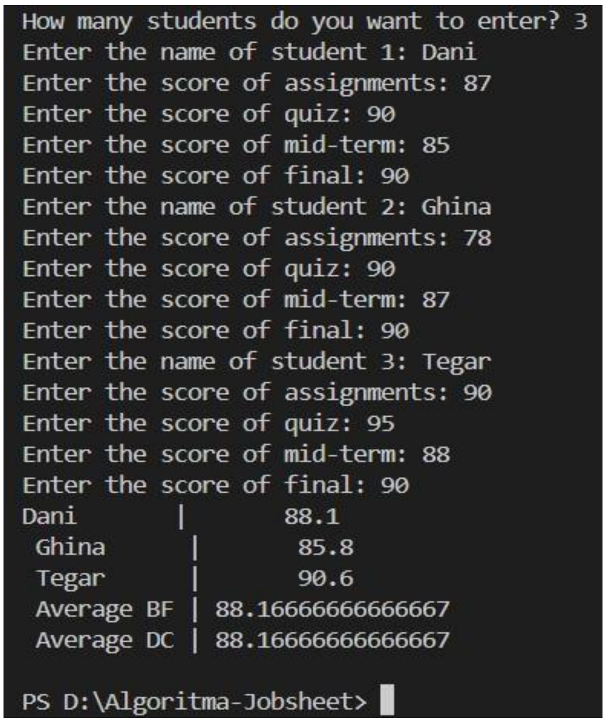
```java
System.out.println(x:"Program for Calculating Total Profits");
System.out.println(x:"Input the Number of Months : ");
int companyCount = sc.nextInt();
Sum[] sm = new Sum[companyCount];

System.out.println(x:"=====================================================");
System.out.print(s:"Input the number of month : ");
int totElemn = sc.nextInt();

for (int i = 0; i < companyCount; i++) {
    sm[i] = new Sum(totElemn);
}

System.out.println(x:"=====================================================");
for (int i = 0; i < sm.length; i++) {
    System.out.println("Company " + (i + 1));
    for (int j = 0; j < sm[0].elemen; j++) {
        System.out.println("Input the profit of the month to - " + (i + 1) + " = ");
        sm[i].profit[j] = sc.nextDouble();
    }
}

System.out.println(x:"=====================================================");
System.out.println(x:"Algorithm brute force");
for (int i = 0; i < sm.length; i++) {
    System.out.println("Total profits of company " + (i + 1) + " for " + sm[i].elemen + " month is = "
            + sm[i].totalBF(sm[i].profit));
}

System.out.println(x:"=====================================================");
System.out.println(x:"Algorithm divide and conquer");
for (int i = 0; i < sm.length; i++) {
    System.out.println("Total profits of company " + (i + 1) + " for " + sm[i].elemen + " month is = "
            + sm[i].totalDC(sm[i].profit, l:0, sm[i].elemen - 1));
```



```
Program for Calculating Total Profits
Input how many company you want to calculate :
2
=====================================================
Input the number of month : 2
=====================================================
Company 1
Input the profit of the month to - 1 =
1000
Input the profit of the month to - 1 =
4000
Company 2
Input the profit of the month to - 2 =
2000
Input the profit of the month to - 2 =
3000
=====================================================
Algorithm brute force
Total profits of company 1 for 2 month is = 5000.0
Total profits of company 2 for 2 month is = 5000.0
=====================================================
Algorithm divide and conquer
Total profits of company 1 for 2 month is = 5000.0
Total profits of company 2 for 2 month is = 5000.0
PS D:\Algoritma-Jobsheet> █
```

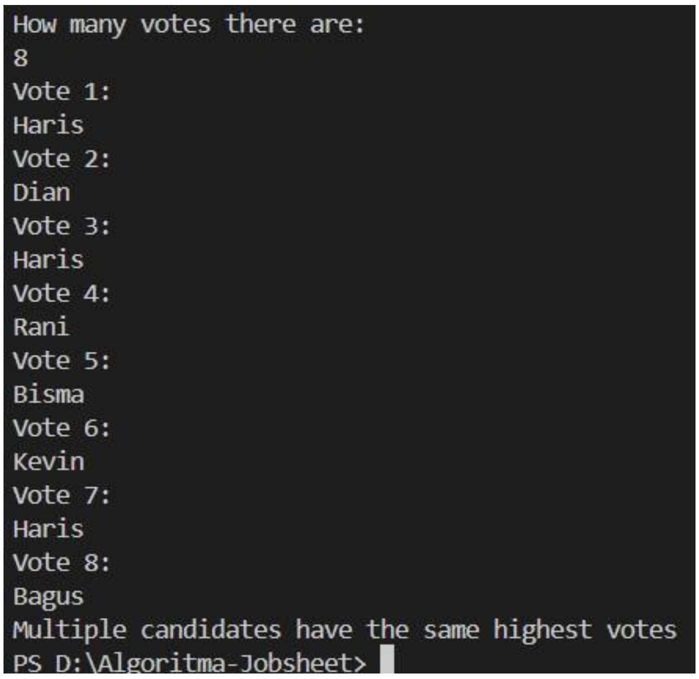
2.6 ASSIGNMENTS

1. Code:
   https://github.com/akhunzakp/semester-2/blob/main/jobsheet5/Assignment1.java
   Output:



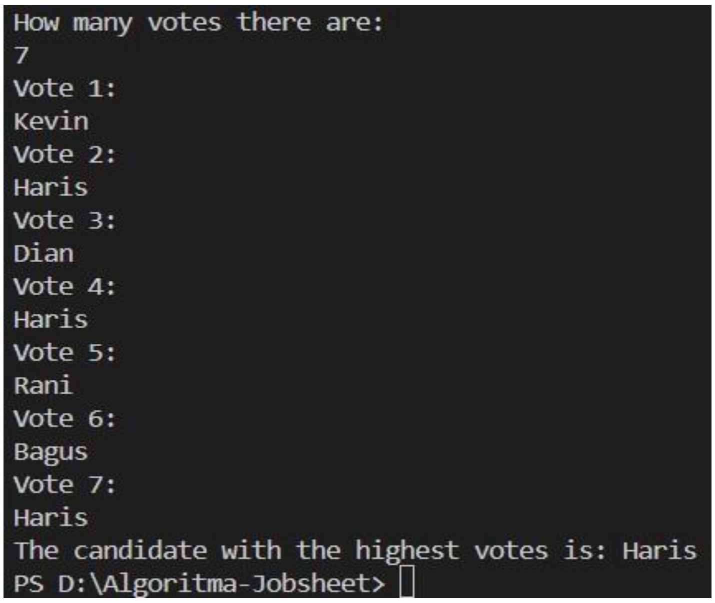
```
How many students do you want to enter? 3
Enter the name of student 1: Dani
Enter the score of assignments: 87
Enter the score of quiz: 90
Enter the score of mid-term: 85
Enter the score of final: 90
Enter the name of student 2: Ghina
Enter the score of assignments: 78
Enter the score of quiz: 90
Enter the score of mid-term: 87
Enter the score of final: 90
Enter the name of student 3: Tegar
Enter the score of assignments: 90
Enter the score of quiz: 95
Enter the score of mid-term: 88
Enter the score of final: 90
Dani        |       88.1
 Ghina      |       85.8
 Tegar      |       90.6
 Average BF | 88.16666666666667
 Average DC | 88.16666666666667

PS D:\Algoritma-Jobsheet>
```

2, Code:
https://github.com/akhunzakp/semester-2/blob/main/jobsheet5/Assignment2.java
Output:



```
How many votes there are:
8
Vote 1:
Haris
Vote 2:
Dian
Vote 3:
Haris
Vote 4:
Rani
Vote 5:
Bisma
Vote 6:
Kevin
Vote 7:
Haris
Vote 8:
Bagus
Multiple candidates have the same highest votes
PS D:\Algoritma-Jobsheet>
```

What if the number of votes is odd? should there be a program improvement? if yes, improve the program for case study no 4. If the number of votes collected is not always even!



```
How many votes there are:
7
Vote 1:
Kevin
Vote 2:
Haris
Vote 3:
Dian
Vote 4:
Haris
Vote 5:
Rani
Vote 6:
Bagus
Vote 7:
Haris
The candidate with the highest votes is: Haris
PS D:\Algoritma-Jobsheet> 
```

No, because my program is already been optimisized to handle odd numbers