JOBSHEET 5

Brute Force and Divide Conquer

Name
Sherly Lutfi Azkiah Sulistyawati

NIM
2341720241

Class
1I

Major
Information Technology

Study Program
D4 Informatics Engineering

Lab Activity 1: Calculating Factorial Values with Brute Force and Divide and Conquer Algorithms

Result

Practice > Week5 > MainFaktorial.java > MainFaktorial

```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 elements 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 Result 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 Result with Divied 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 elements 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 Result with Brute Force
Factorial of value 5 is : 120
Factorial of value 8 is : 40320
Factorial of value 3 is : 6
=====================================================
Factorial Result with Divied and Conquer
Factorial of value 5 is : 120
Factorial of value 8 is : 40320
Factorial of value 3 is : 6
=====================================================
```

Question

1. Explain the Divide Conquer Algorithm for calculating factorial values!
   - The Divide Conquer algorithm for calculating factorial works by breaking down the problem into smaller subproblems until they become simple enough to solve directly.

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!
   - In the given implementation, the Divide and Conquer algorithm for calculating factorial is represented by the faktorialDC(int n) method. Let's break down each part:
     - Divide: This part is implicit in the recursive nature of the method. Each recursive call divides the problem into a smaller subproblem by decrementing n.
     - Conquer: The faktorialDC(int n) method recursively calls itself with n-1 until n becomes 1. At this point, it returns 1, which is the base case.
     - Combine: There is no explicit combine step in this implementation. However, the combination of results happens implicitly as the recursive calls return and compute the final result.
3. Is it possible to repeat the factorial BF () method instead of using for? Prove it!
   - Yes, it's possible to repeat the faktorialBF() method without using a for loop. We can achieve this by using recursion.

```
public int faktorialBF(int n) {
    if (n==1) {
        return 1;
    } else {
        return n *faktorialBF(n-1);
    }
}
```

```
=========================================================
Input the number of elements 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 Result with Brute Force
Factorial of value 5 is : 120
Factorial of value 8 is : 40320
Factorial of value 3 is : 6
=========================================================
Factorial Result with Divied and Conquer
Factorial of value 5 is : 120
Factorial of value 8 is : 40320
Factorial of value 3 is : 6
=========================================================
```

4. Add a check to the execution time of the two types of methods!
   - Add timing checks to measure the execution time of each method using System.currentTimeMillis() or System.nanoTime().

```
System.out.println(x:"=====================================================");
System.out.println(x:"Factorial Result with Brute Force");
for (int i = 0; i < elemen; i++) {
    long startTime = System.nanoTime();
    int resultBF = fk[i].faktorialBF(fk[i].num);
    long endTime = System.nanoTime();
    long durationBF = (endTime - startTime);  // Time in nanoseconds
    System.out.println("Factorial of value " + fk[i].num + " is : " + fk[i].faktorialBF(fk[i].num));
    System.out.println("Execution time (Brute Force): " + durationBF + " nanoseconds");
}

System.out.println(x:"=====================================================");
System.out.println(x:"Factorial Result with Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    long startTime = System.nanoTime();
    int resultDC = fk[i].faktorialDC(fk[i].num);
    long endTime = System.nanoTime();
    long durationDC = (endTime - startTime);  // Time in nanoseconds
    System.out.println("Factorial of value " + fk[i].num + " is : " + fk[i].faktorialDC(fk[i].num));
    System.out.println("Execution time (Divide and Conquer): " + durationDC + " nanoseconds");
}
System.out.println(x:("=====================================================");
```

```
=====================================================
Input the number of elements 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 Result with Brute Force
Factorial of value 5 is : 120
Execution time (Brute Force): 19300 nanoseconds
Factorial of value 8 is : 40320
Execution time (Brute Force): 3100 nanoseconds
Factorial of value 3 is : 6
Execution time (Brute Force): 1800 nanoseconds
=====================================================
Factorial Result with Divide and Conquer
Factorial of value 5 is : 120
Execution time (Divide and Conquer): 71600 nanoseconds
Factorial of value 8 is : 40320
Execution time (Divide and Conquer): 2200 nanoseconds
Factorial of value 3 is : 6
Execution time (Divide and Conquer): 1500 nanoseconds
=====================================================
```

5. Prove by inputting elements that are above 20 digits, is there a difference in execution time?
   - Yes, there will likely be a noticeable difference in execution time for larger input values due to the nature of the algorithms. The brute-force method (faktorialBF()) has a time complexity of O(n) because it iterates through all numbers from 1 to n. However, the divide and conquer method (faktorialDC()) has a time complexity of O(log n) due to its recursive nature. As the input value increases, the difference in

execution time between the two methods will become more apparent, with the divide and conquer method being more efficient for larger inputs.

Lab Activity 2: Calculating Squared Results with Brute Force and Divide and Conquer Algorithms

Result

Practice > Week5 > MainSquared.java > MainSquared > main(String[])

```java
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 square value to-" + (i+1) + " : ");
            png[i].squared = sc.nextInt();
        }

        System.out.println(x:"=====================================================");
        System.out.println(x:"Result with Brute Force Squared");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Value " + png[i].num + " squared " + png[i].squared + " is : "
            + png[i].squareBF(png[i].num, png[i].squared));
        }
        System.out.println(x:"=====================================================");
        System.out.println(x:"Result with Divied and Conquer Squared");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Value " + png[i].num + " squared " + png[i].squared + " is : "
            + png[i].squareDC(png[i].num, png[i].squared));
        }
        System.out.println(x:"=====================================================");

        sc.close();
    }
}
```

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

Question

1. Explain the differences between the 2 methods made are SquaredBF() and SquaredDC()!
   - SquaredBF(): This method uses a brute-force approach to calculate the power of a number. It iteratively multiplies the base number by itself n times.
     SquaredDC(): This method utilizes a divide and conquer approach. It recursively divides the power n by 2 until it reaches the base case (n == 0), and then combines the results accordingly. If n is odd, it splits the power into two equal parts and multiplies them by the base number and then multiplies the result by the base number again. If n is even, it splits the power into two equal parts and multiplies them 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!
   - This part of the code is handling the recursive calculation in the divide and conquer approach. It checks if the power n is odd or even.
     - If n is odd, it splits the power into two equal parts (n/2), recursively calculates the square of the base number for both parts, and then multiplies them together. Finally, it multiplies the result by the base number a.
     - If n is even, it splits the power into two equal parts (n/2), recursively calculates the square of the base number for both parts, and then multiplies them together without multiplying by a again.
3. Explain whether the combine stage is included in the code!
   - The combine stage in a divide and conquer algorithm is essentially the part where the results of the recursive calls are combined to produce the final result. In the provided SquaredDC() method, the combine stage is implicit in the way the recursive calls are made and the results are multiplied together. Each recursive call returns a result, and these results are multiplied together in the return statement, which effectively combines them to produce the final squared value.
4. Modification of the program code, assuming the attribute filling process is done by a constructor.

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

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

```
27          switch (choice) {
28              case 1:
29                  System.out.println(x:"=====================================================");
30                  System.out.println(x:"Result with Brute Force Squared");
31                  for (int i = 0; i < elemen; i++) {
32                      System.out.println("Value " + png[i].num + " squared " + png[i].squared + " is : "
33                      + png[i].squareBF(png[i].num, png[i].squared));
34                  }
35                  break;
36
37              case 2:
38                  System.out.println(x:"=====================================================");
39                  System.out.println(x:"Result with Divied and Conquer Squared");
40                  for (int i = 0; i < elemen; i++) {
41                      System.out.println("Value " + png[i].num + " squared " + png[i].squared + " is : "
42                      + png[i].squareDC(png[i].num, png[i].squared));
43                  }
44                  break;
45
46              default:
47                  System.out.println(x:"=====================================================");
48                  System.out.println(x:"Invalid choice.");
49                  break;
50          }
```

```
Select the method to run:
1. Brute Force Squred
2. Divide and Conquer Squared
1
=====================================================
Input the number of elements you want to count : 2
Input the value to be squared to-1 : 4
Input the square value to-1 : 6
Input the value to be squared to-2 : 5
Input the square value to-2 : 3
=====================================================
Result with Brute Force Squared
Value 4 squared 6 is : 4096
Value 5 squared 3 is : 125
=====================================================
```

```
Select the method to run:
1. Brute Force Squred
2. Divide and Conquer Squared
2
=====================================================
Input the number of elements you want to count : 2
Input the value to be squared to-1 : 4
Input the square value to-1 : 6
Input the value to be squared to-2 : 5
Input the square value to-2 : 3
=====================================================
Result with Divied and Conquer Squared
Value 4 squared 6 is : 4096
Value 5 squared 3 is : 125
=====================================================
```

Lab Activity 3: Calculating Sum Array with Brute Force and Divide and Conquer Algorithms

Result

```
=========================================================
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
```

Question

1. Give an illustration of the difference in profit calculation with the TotalBF () or TotalDC () method.
   - TotalBF(): This method calculates the total profit by simply iterating over the array of profits and summing them up. It has a time complexity of O(n), where n is the number of months.

     TotalDC(): This method uses a divide and conquer approach to calculate the total profit. It recursively divides the array into halves until individual elements are reached, then combines the results. It has a time complexity of O(n log n) due to its divide and conquer nature.

     Illustration:
     - Suppose we have profits for 12 months: [1000, 2000, 1500, 3000, 2500, 1800, 4000, 3500, 2700, 2200, 3200, 2800].
     - TotalBF() will simply sum all these values: 1000 + 2000 + 1500 + ... + 2800 = 29800.
     - TotalDC() will recursively divide the array into halves until it calculates the total profit for each half and combines them. It's a bit more complex but achieves the same result.
2. Why is there the following return value? Explain!
   - The return statement in the totalDC() method is used to return the sum of profits calculated for the left subarray (lsum), the right subarray (rsum), and the middle element (arr[mid]). This is a crucial part of the divide and conquer approach. It ensures that the total profit is correctly calculated by summing up the profits from both halves and adding the profit from the middle element.
3. Why is the mid variable required for the TotalDC () method?
   - The mid variable is required to divide the array into two halves in each recursive call. It determines the middle index of the array, which separates the array into two subarrays - one from l to mid-1, and the other from mid+1 to r. This splitting is essential for the divide and conquer approach to work correctly.
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
public class MainSum {
    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.print(s:"Input the Number of Companies: ");
        int numCompanies = sc.nextInt();

        Sum[] companies = new Sum[numCompanies];

        System.out.println(x:"=====================================================");
        for (int i = 0; i < numCompanies; i++) {
            System.out.print("Input the number of months for company " + (i+1) + ": ");
            int months = sc.nextInt();
            companies[i] = new Sum(months);

            System.out.println("Input profits for company " + (i+1) + " : ");
            for (int j = 0; j < months; j++) {
                System.out.print("Profit for month " + (j+1) + " = ");
                companies[i].profit[j] = sc.nextDouble();
            }
        }

        System.out.println(x:"=====================================================");
        for (int i = 0; i < companies.length; i++) {
            System.out.println("Company " + (i+1) + " - Total profits:");
            System.out.println("Algoritma Brute Force = " + companies[i].totalBF(companies[i].profit));
            System.out.println("Algoritma Divide Conquer = " + companies[i].totalDC(companies[i].profit, l:0, companies[i].elemen-1));
        }
    }
```

```
=====================================================
Program for Calculating Total Profits
Input the Number of Companies: 2
=====================================================
Input the number of months for company 1: 2
Input profits for company 1 :
Profit for month 1 = 2000
Profit for month 2 = 3000
Input the number of months for company 2: 3
Input profits for company 2 :
Profit for month 1 = 3000
Profit for month 2 = 2500
Profit for month 3 = 4000
=====================================================
Company 1 - Total profits:
Algoritma Brute Force = 5000.0
Algoritma Divide Conquer = 5000.0
Company 2 - Total profits:
Algoritma Brute Force = 9500.0
Algoritma Divide Conquer = 9500.0
=====================================================
```

Assignment

1. Create a program based on the following class diagram!

ScoreAlgSdt
nameSdt: String scoreAssgment: int scoreQuiz: int scoreMid: int scoreFinal: int
CalculateTotalScore(): double

Score are calculated based on total Assignment of 30%, Quiz 20%, Mid 20%, Final 30%. Adjust the method if it must have parameters.

Practice > Week5 > ScoreAlgSdt.java > ScoreAlgSdt

```java
package Week5;

public class ScoreAlgSdt {
    String nameSdt;
    int scoreAssignment;
    int scoreQuiz;
    int scoreMid;
    int scoreFinal;

    ScoreAlgSdt(String nameSdt, int scoreAssignment, int scoreQuiz, int scoreMid, int scoreFinal) {
        this.nameSdt = nameSdt;
        this.scoreAssignment = scoreAssignment;
        this.scoreQuiz = scoreQuiz;
        this.scoreMid = scoreMid;
        this.scoreFinal = scoreFinal;
    }

    double calculateTotalScore() {
        double totalScore = (scoreAssignment * 0.3) + (scoreQuiz * 0.2) + (scoreMid * 0.2) + (scoreFinal * 0.3);
        return totalScore;
    }

    String getNameSdt() {
        return nameSdt;
    }
}
```

2. (continued about question no. 1). Create an array of objects in the main class to find out the value of several students in one calculation at a time!

Practice > Week5 > MainScoreAlgSdt.java > MainScoreAlgSdt

```java
package Week5;

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

        ScoreAlgSdt[] students = new ScoreAlgSdt[3];
        students[0] = new ScoreAlgSdt(nameSdt:"Rani", scoreAssignment:90, scoreQuiz:80, scoreMid:85, …95);
        students[1] = new ScoreAlgSdt(nameSdt:"Dani", scoreAssignment:70, scoreQuiz:75, scoreMid:80, …85);
        students[2] = new ScoreAlgSdt(nameSdt:"Saraswati", scoreAssignment:85, scoreQuiz:90, scoreMid:95, …85);
```

3. (Continued question no. 2) with the modified Brute Force algorithm program in order to know the average value of all students who have been inputted to the Algorithm and data structure course.

   Example :

   There are a total of the following student grades

Name	Total Value of Algorithm Courses
Rani	86,5
Dani	72,5
Saraswati	89
Average	82,67

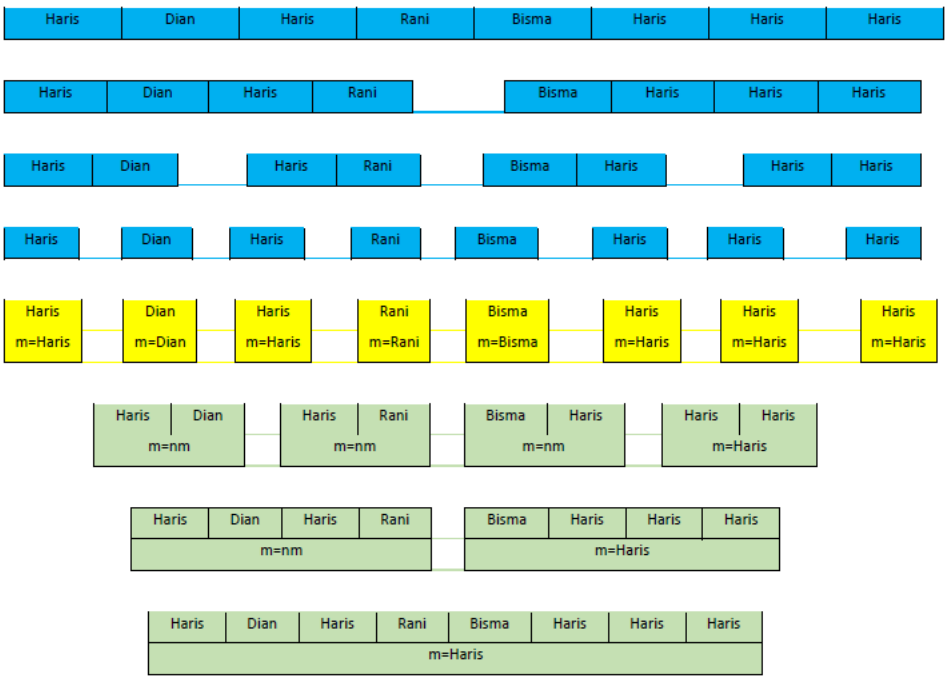
   Then the average total value of all students who have taken the algorithm course is 82.67

```
Practice > Week5 > MainScoreAlgSdt.java > MainScoreAlgSdt
1  package Week5;
2
3  public class MainScoreAlgSdt {
       Run | Debug
4      public static void main(String[] args) {
5
6          ScoreAlgSdt[] students = new ScoreAlgSdt[3];
7          students[0] = new ScoreAlgSdt(nameSdt:"Rani", scoreAssignment:90, scoreQuiz:80, scoreMid:85, …95);
8          students[1] = new ScoreAlgSdt(nameSdt:"Dani", scoreAssignment:70, scoreQuiz:75, scoreMid:80, …85);
9          students[2] = new ScoreAlgSdt(nameSdt:"Saraswati", scoreAssignment:85, scoreQuiz:90, scoreMid:95, …85);
10
11         double total = 0;
12         System.out.println(x:"Name\t\tTotal Value of Algorithm Courses");
13         System.out.println(x:"================================================");
14         for (ScoreAlgSdt student : students) {
15             double totalScore = student.calculateTotalScore();
16             System.out.println(student.getNameSdt() + "\t\t" + totalScore);
17             total += totalScore;
18         }
19         double average = total / students.length;
20         System.out.println("Average\t\t" + average);
```

```
Name            Total Value of Algorithm Courses
================================================
Rani            88.5
Dani            77.5
Saraswati               88.0
Average         84.66666666666667
```

4. A university in Malang is holding a vote to elect the BEM chairman in 2020. If the number of votes collected is always even. Then by inputting the selected candidates, count the number of votes for each candidate. Make class diagrams and programs using the Divide and Conquer algorithm from the case study! (The number of array elements and the results of the vote are user input)

   Example: Voting results are as follows (m is majority, nm is not majority)



   Description: Blue is the divide process, yellow starts the conquer process, green starts the combining process

Practice > Week5 > Vote.java > Vote > main(String[])

```java
package Week5;

import java.util.Scanner;

public class Vote {
    public static String countVotes(String[] votes, int start, int end) {
        if (start == end) {
            // System.out.println("Start: " + votes[start]);
            return votes[start];
        } else {
            int mid = start + (end - start) / 2;
            // System.out.println("mid: " + mid);
            String leftCandidates = countVotes(votes, start, mid);
            String rightCandidates = countVotes(votes, mid + 1, end);
            // System.out.println("left candidates: " + leftCandidates);
            // System.out.println("right candidates: " + rightCandidates);

            if (leftCandidates.equals(rightCandidates)) {
                // System.out.println("left candidates akhir: " + leftCandidates);
                return leftCandidates;
            } else {
                int leftCount = countVotesHelper(votes, leftCandidates, start, end);
                int rightCount = countVotesHelper(votes, rightCandidates, start, end);
                // System.out.println("left count: " + leftCount);
                // System.out.println("right count: " + rightCount);
                if (leftCount > rightCount) {
                    // System.out.println("candidate left: " + leftCandidates);
                    return leftCandidates;
                } else if (rightCount > leftCount) {
                    // System.out.println("candidate right: " + rightCandidates);
                    return rightCandidates;
                } else {
                    return "m";
                }

            }
    public static int countVotesHelper(String[] votes, String candidate, int start, int end) {
        int count = 0;
        // System.out.println("start: " + start);
        // System.out.println("end: " + end);
        for (int i = start; i <= end; i++) {
            if (votes[i].equals(candidate)) {
                count++;
            }
        }

        return count;
    }
    static Scanner sc = new Scanner(System.in);
    Run | Debug
    public static void main(String[] args) {


        System.out.print(s:"Enter the number of votes: ");
        int numVotes = sc.nextInt();

        String[] votes = new String[numVotes];

        for (int i = 0; i < votes.length; i++) {
            System.out.print("Vote " + (i + 1) + ": ");
            votes[i] = sc.next();
        }

        // String votes[] = {"A", "B", "A", "C", "B", "A", "A"};
        // // String votes[] = {"A", "B", "B", "A"};

        String highestCandidate = countVotes(votes, start:0, votes.length - 1);

        if (highestCandidate.equals(anObject:"m")) {
            System.out.println(x:"Multiple candidates have the same highest votes");
        } else {
            System.out.println("The candidate with the highest votes is: " + highestCandidate);
```

```
Enter the number of votes: 8
Vote 1: Haris
Vote 2: Dian
Vote 3: Haris
Vote 4: Rani
Vote 5: Bisma
Vote 6: Haris
Vote 7: Haris
Vote 8: Haris
The candidate with the highest votes is: Haris
```

5. 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!

```
Practice > Week5 > Vote.java > Vote > main(String[])
package Week5;

import java.util.Scanner;

public class Vote {
    public static String countVotes(String[] votes, int start, int end) {
        if (start == end) {
            // System.out.println("Start: " + votes[start]);
            return votes[start];
        } else {
            int mid = start + (end - start) / 2;
            // System.out.println("mid: " + mid);
            String leftCandidates = countVotes(votes, start, mid);
            String rightCandidates = countVotes(votes, mid + 1, end);
            // System.out.println("left candidates: " + leftCandidates);
            // System.out.println("right candidates: " + rightCandidates);

            if (leftCandidates.equals(rightCandidates)) {
                // System.out.println("left candidates akhir: " + leftCandidates);
                return leftCandidates;
            } else {
                int leftCount = countVotesHelper(votes, leftCandidates, start, end);
                int rightCount = countVotesHelper(votes, rightCandidates, start, end);
                // System.out.println("left count: " + leftCount);
                // System.out.println("right count: " + rightCount);
                if (leftCount > rightCount) {
                    // System.out.println("candidate left: " + leftCandidates);
                    return leftCandidates;
                } else if (rightCount > leftCount) {
                    // System.out.println("candidate right: " + rightCandidates);
                    return rightCandidates;
                } else {
                    return "m";
                }

            }
```

```
    public static int countVotesHelper(String[] votes, String candidate, int start, int end) {
        int count = 0;
        // System.out.println("start: " + start);
        // System.out.println("end: " + end);
        for (int i = start; i <= end; i++) {
            if (votes[i].equals(candidate)) {
                count++;
            }
        }

        return count;
    }
    static Scanner sc = new Scanner(System.in);
    Run | Debug
    public static void main(String[] args) {


        System.out.print(s:"Enter the number of votes: ");
        int numVotes = sc.nextInt();

        String[] votes = new String[numVotes];

        for (int i = 0; i < votes.length; i++) {
            System.out.print("Vote " + (i + 1) + ": ");
            votes[i] = sc.next();
        }

        // String votes[] = {"A", "B", "A", "C", "B", "A", "A"};
        // // String votes[] = {"A", "B", "B", "A"};

        String highestCandidate = countVotes(votes, start:0, votes.length - 1);

        if (highestCandidate.equals(anObject:"m")) {
            System.out.println(x:"Multiple candidates have the same highest votes");
        } else {
            System.out.println("The candidate with the highest votes is: " + highestCandidate);
```

6. Modify the program about the average value of algorithm courses using the Divide and Conquer Algorithm!

```
Practice > Week5 > Vote.java > Vote > main(String[])
1   package Week5;
2
3   import java.util.Scanner;
4
5   public class Vote {
6       public static String countVotes(String[] votes, int start, int end) {
7           if (start == end) {
8               // System.out.println("Start: " + votes[start]);
9               return votes[start];
10          } else {
11              int mid = start + (end - start) / 2;
12              // System.out.println("mid: " + mid);
13              String leftCandidates = countVotes(votes, start, mid);
14              String rightCandidates = countVotes(votes, mid + 1, end);
15              // System.out.println("left candidates: " + leftCandidates);
16              // System.out.println("right candidates: " + rightCandidates);
17
18              if (leftCandidates.equals(rightCandidates)) {
19                  // System.out.println("left candidates akhir: " + leftCandidates);
20                  return leftCandidates;
21              } else {
22                  int leftCount = countVotesHelper(votes, leftCandidates, start, end);
23                  int rightCount = countVotesHelper(votes, rightCandidates, start, end);
24                  // System.out.println("left count: " + leftCount);
25                  // System.out.println("right count: " + rightCount);
26                  if (leftCount > rightCount) {
27                      // System.out.println("candidate left: " + leftCandidates);
28                      return leftCandidates;
29                  } else if (rightCount > leftCount) {
30                      // System.out.println("candidate right: " + rightCandidates);
31                      return rightCandidates;
32                  } else {
33                      return "m";
34                  }
35
36              }
40      public static int countVotesHelper(String[] votes, String candidate, int start, int end) {
41          int count = 0;
42          // System.out.println("start: " + start);
43          // System.out.println("end: " + end);
44          for (int i = start; i <= end; i++) {
45              if (votes[i].equals(candidate)) {
46                  count++;
47              }
48          }
49
50          return count;
51      }
52      static Scanner sc = new Scanner(System.in);
        Run | Debug
53      public static void main(String[] args) {
54
55
56          System.out.print(s:"Enter the number of votes: ");
57          int numVotes = sc.nextInt();
58
59          String[] votes = new String[numVotes];
60
61          for (int i = 0; i < votes.length; i++) {
62              System.out.print("Vote " + (i + 1) + ": ");
63              votes[i] = sc.next();
64          }
65
66          // String votes[] = {"A", "B", "A", "C", "B", "A", "A"};
67          // // String votes[] = {"A", "B", "B", "A"};
68
69          String highestCandidate = countVotes(votes, start:0, votes.length - 1);
70
71          if (highestCandidate.equals(anObject:"m")) {
72              System.out.println(x:"Multiple candidates have the same highest votes");
73          } else {
74              System.out.println("The candidate with the highest votes is: " + highestCandidate);
```