Christopher Bussen

CPS 150 02 – Algorithms and Programming 1

Long Week Project 3

11/26/20

**Problem 1 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the integers for the first array
4. Declare an int array variable with 11 elements and use a for loop to enter the user input
5. Prompt the user to enter the integers for the second array
6. Declare another int array variable with 11 elements and use a for loop to enter the user input
7. Create a separate method that checks if the arrays can be the same – method should take two int arrays as input and return a Boolean
8. Check to see if the arrays are the same length – if they aren’t return false
9. Sort both arrays
10. Use a for loop to check if each element in the first array is equal to the corresponding element in the second array
11. If one of the corresponding elements in the second array is not equal to the element in the first, then return false
12. If the for loop ends without returning false, return true at the end of the method
13. End the method that checks the arrays
14. Call the method that checks the arrays in an if statement – use the first and second array as inputs in the method
15. If the method returns true, then print YES – otherwise print NO
16. End main method
17. End program

**Problem 1 Running Screenshot**

**Text

Description automatically generated**

**Problem 1 Code**

/\*

Christopher Bussen

CPS 150 02

Long Week Project 3

CheckEqualArrays: int array int array ; string

program takes in two 11 integer array elements and determines if there

is a permutation of the first array that equals the second array - if

so, print yes, otherwise print no

ex1: user inputs [1 2 3 4 5 6 7 8 9 10 11], [11 1 2 10 9 3 4 8 7 5 6] - program outputs YES

ex2: user inputs [1 2 5 3 7 0 7 3 5 2 1], [7 3 1 2 5 0 5 2 1 3 7] - program outputs YES

ex3: user inputs [a b c d e f g h i j k], [a b c d e f g h i j k] - program outputs error

ex4: user inputs [1.5 2 3 4 5 6 7 8 9 10 11], [1.5 2 3 4 5 6 7 8 9 10 11] - program outputs error

ex5: user inputs [1 2 3 4 5 6 7 8 9 10 11], [6 5 4 3 2 0 11 10 9 8 7] - program outputs NO

\*/

import java.util.Arrays;

import java.util.Scanner;

public class CheckEqualArrays {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt the user to enter the integers for the first array

System.out.print("Enter 11 integers for the first array: ");

//declare an int array variable with 11 elements and use a for loop to enter the user input

int[] array1 = new int [11];

for(int i = 0; i < 11; i++){

array1[i] = input.nextInt();

}

//prompt the user to enter the integers for the second array

System.out.print("Enter 11 integers for the second array: ");

//declare another int array variable with 11 elements and use a for loop to enter the user input

int[] array2 = new int [11];

for(int i = 0; i < 11; i++){

array2[i] = input.nextInt();

}

//call the method the checkArrays method in an if statement - use two user input arrays as inputs

//if the method returns true, print yes, otherwise print no

if(checkArrays(array1, array2)){

System.out.println("YES");

}

else{

System.out.println("NO");

}

}

//create separate method that checks if arrays can be the same - takes two int arrays as input, returns boolean

public static boolean checkArrays(int[] a, int[] b){

//check to see if the arrays are the same length - if not, return false

if(a.length != b.length){

return false;

}

//sort the arrays

Arrays.sort(a);

Arrays.sort(b);

//use for loop to check if all elements in both arrays are equal - if not, return false

for(int x = 0; x < a.length; x++){

if(a[x] != b[x]){

return false;

}

}

//return true if the for loop ends without returning false

return true;

}

}

**Problem 2 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the index of the new start of the array and declare an int variable for this value
4. Declare an int array variable for the array given in the problem
5. Use nested for loops to split the array at the index given by the user
6. In the outer for loop, declare an int variable for the first element of the array
7. In the inner for loop, move all elements after the first element up 1
8. In the outer for loop, add the first element to the end of the array
9. Repeat this loop n times, where n is the int value entered by the user for the new start index
10. Create another for loop to print all of the elements in the new array
11. End the program

**Problem 2 Running Screenshot**

**Text

Description automatically generated**

**Problem 2 Code**

/\*

Christopher Bussen

CPS 150 02

Long Week Project 3

SplitArray: number ; string

program takes in an int from the user that represents the new index to start

the array from and moves everything before that index to the end of the array

then prints the new array

array = {12, 10, 5, 6, 52, 36}

ex1: user inputs 2 - program outputs 5, 6, 52, 36, 12, 10

ex2: user inputs 1 - program outputs 10, 5, 6, 52, 36, 12

ex3: user inputs 6 - program outputs 12, 10, 5, 6, 52, 36

ex4: user inputs 1.5 - program outputs error

ex5: user inputs x - program outputs error

\*/

import java.util.Scanner;

public class SplitArray {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt user to enter index of new start and declare an int variable for it

System.out.print("Enter the index of the new start of the array: ");

int newStart = input.nextInt();

//declare an int array variable for the array given in the problem

int[] arr = {12, 10, 5, 6, 52, 36};

//use nested for loops to split the array at the given index

for(int i = 0; i < newStart; i++){

//declare an int variable for the first element in the array

int x = arr[0];

for(int j = 0; j < arr.length-1; j++){

//move all elements after the first element up one

arr[j] = arr[j+1];

}

//add the first element onto the end of the array

arr[arr.length-1] = x;

}

//use for loop to print the array

for(int k = 0; k < arr.length; k++){

System.out.print(arr[k] + ", ");

}

}

}

**Problem 3 Algorithm**

1. Start the program
2. Import the scanner
3. Declare an int array variable with 8 elements for the user’s values
4. Prompt the user to enter values for the array and use a for loop to add them to the array
5. Create a separate method to determine the maximum contiguous sum in the array – should take in an int array and return an int
6. Declare an int variable for the max so far
7. Declare an int variable for the max ending after a specific element
8. Use a for loop to go through each element and find the maximum contiguous sum \
9. In the for loop, add the array element to the maxEndingHere variable
10. If maxEndingHere is greater than maxSoFar, set maxSoFar equal to maxEndingHere
11. If maxEndingHere is less than 0, set it equal to 0
12. Return maxSoFar variable
13. End the method that determines the maximum contiguous sum
14. Print the maximum contiguous sum by calling the method using the array with the user’s values
15. End the main method
16. End the program

**Problem 3 Running Screenshot**

**Text

Description automatically generated**

**Problem 3 Code**

/\*

Christopher Bussen

CPS 150 02

Long Week Project 3

LargestSubarray: number number number number number number number number ; number

program takes 8 numbers from the user to be used in an int array and

determines and prints out the maximum contiguous sum within the array

ex1: user inputs -2, -3, 4, -1, -2, 1, 5, -3 - program outputs 7

ex2: user inputs 1, 2, 3, 4, -1, -8, 5 - program outputs 10

ex3: user inputs -1, -2, -3, -4, -5, -6, -7, -8 - program outputs 0

ex4: user inputs 1.5, 2, 4, -10, 6, 91, 0, 1 - program outputs error

ex5: user inputs x, y, z, a, b, salt, mom, dad - program outputs error

\*/

import java.util.Scanner;

public class LargestSubarray {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//declare an int array variable with 8 elements for the user's values

int[] userValues = new int[8];

//prompt the user to enter values for the array and use a for loop to add them to the array

System.out.print("Enter 8 integers for the array: ");

for(int i = 0; i < userValues.length; i++){

userValues[i] = input.nextInt();

}

//print max - call method that finds the maximum contiguous string using the array with user values

System.out.println("The maximum contiguous sum is " + maximumSum(userValues));

}

//create a separate method to determine the largest contiguous sum - takes in an int array and returns an int

public static int maximumSum(int[] a){

//declare an int variable for the max so far

int maxSoFar = 0;

//declare an int variable for the max ending after a specific element

int maxEndingHere = 0;

//use a for loop to go through each element of the array and find the max contiguous sum

for(int j = 0; j < a.length; j++){

maxEndingHere = maxEndingHere + a[j];

if(maxSoFar < maxEndingHere){

maxSoFar = maxEndingHere;

}

if(maxEndingHere < 0){

maxEndingHere = 0;

}

}

return maxSoFar;

}

}

**Problem 4 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the length of the array and declare an int variable for it
4. Declare an int array variable to hold the user’s values
5. Prompt the user to enter the values for the array and use a for loop to enter them into the array
6. Sort the array
7. Create a separate method to remove duplicates from the array – this method should take in an int array and return an int
8. If the array’s length is less than 2, return the array’s length
9. Use a for loop to delete any duplicates in the array – if one element is the same as the one behind it, delete the element
10. Print the new array using a for loop
11. Print a new line
12. Return the new array’s length
13. End the method that deletes duplicates
14. Print the new array’s length by calling the method that removes duplicates
15. End the main method
16. End the program

**Problem 4 Running ScreenshotText

Description automatically generated**

**Problem 4 Code**

/\*

Christopher Bussen

CPS 150 02

Long Week Project 3

Remove duplicates: numbers (amount of inputs depends on first number) ; number

program takes in a number for the array length from the user then takes in that many

numbers as inputs for the array - program then gets rid of any duplicate values, prints

the new array, and returns/prints the new length of the array

ex1: user inputs 3, 1, 1, 2 - program outputs 2

ex2: user inputs 10, 0, 0, 1, 1, 1, 2, 2, 3, 3, 4 - program outputs 5

ex3: user inputs 7, 0, 12, 0, 7, 0, 7, 3 - program outputs 4

ex4: user inputs 5, 2.5, 4, -10, 6, 91 - program outputs error

ex5: user inputs 4, y, z, cookie, remote - program outputs error

\*/

import java.util.Arrays;

import java.util.Scanner;

public class RemoveDuplicates {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt the user to enter the length of the array and declare an int variable for it

System.out.print("Enter the length of the array: ");

int arrLength = input.nextInt();

//declare an int array variable to hold the user's values

int[] arr = new int [arrLength];

//prompt the user to enter the values for the array and use a for loop to enter them in

System.out.print("Enter the array values: ");

for(int i = 0; i < arrLength; i++){

arr[i] = input.nextInt();

}

//sort the array

Arrays.sort(arr);

//print the array's new length by calling the submethod

System.out.println("The array's length without duplicates is " + deleteDupes(arr));

}

//create a separate method to remove duplicates from the array - takes in int array and returns int

public static int deleteDupes(int[] a){

//return the array’s length if the length is less than 2

if(a.length < 2){

return a.length;

}

int k = 0;

//use a for loop to delete the duplicates in the array

for(int j = 1; j < a.length; j++){

if(a[j] != a[k]){

k++;

a[k] = a[j];

}

}

//print the new array using a for loop

for(int x = 0; x < k+1; x++){

System.out.print(a[x] + " ");

}

//print a new line

System.out.println();

//return the new array's length

return k+1;

}

}