Christopher Bussen

CPS 150 02 – Algorithms and Programming 1

Assignment 6

12/11/2020

**Program 1 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the length of the array
4. Declare an int array variable of the length given by the user
5. Prompt the user to enter numbers into the number array
6. Add the user’s values to the array using a for loop
7. Create a separate method to determine if the array creates a palindrome – it should take in an int array and return a Boolean value
8. Declare a new int array of the original length – this will be used to reverse the original array
9. Use a for loop to reverse the original array into the new array
10. Print the reversed array using a for loop and print a new line at the end
11. If the original and reversed arrays are not the same, return false – however, if they are the same, return true
12. End the method that determines if the array is a palindrome and return to the main method
13. Call the method that determines if the array is a palindrome using the original user array as a parameter – if it returns true, tell the user the array is a palindrome – otherwise, tell the user it is not a palindrome
14. End the main method
15. End the program

**Program 1 Running Screenshot**

**Text

Description automatically generated**

**Program 1 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P1CheckForPalindrome: int, int array ; string

program takes in a number from the user for the length of the array as well as numbers to

fill that array then determines and prints out whether the numbers entered is a palindrome

(also prints out reversed array)

ex1: user inputs 8, 4, 6, 3, 5, 5, 3, 6, 4 - program outputs Palindrome = Yes

ex2: user inputs 4, 1, 2, 2, 1 - program outputs Palindrome = Yes

ex3: user inputs 11, 8, 7, 4, 7, 6, 5, 7, 2, 4, 6, 5 - program outputs Palindrome = No

ex4: user inputs 5, 87, 10, 11, 10, 87 - program outputs Palindrome = Yes

ex5: user inputs 5, 1.3, x, y, x, 1.3 - program outputs error

ex6: user inputs 2.0, 12, 12 - program outputs error

\*/

import java.util.Scanner;

public class P1CheckForPalindrome {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt user to enter length of the array

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

int numOfValues = input.nextInt();

//declare an int array variable with the length given by the user

int[] numArray = new int[numOfValues];

//prompt the user to enter values for the array

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

//add the user's values to the array using a for loop

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

numArray[i] = input.nextInt();

}

//call method for determining a palindrome using the number array as a parameter - if it returns true, tell user it's a palindrome - otherwise say it is not

if(isPalindrome(numArray)){

System.out.println("Palindrome = Yes");

}

else{

System.out.println("Palindrome = No");

}

}

//create a separate method to determine if the array is a palindrome - should take in an int array and return a boolean

public static boolean isPalindrome (int[] a) {

//declare a new int array of the same length that will be used to reverse the array

int[] reverseNumArray = new int[a.length];

//use a for loop to reverse the number array

for (int j = a.length - 1; j >= 0; j--) {

reverseNumArray[a.length - 1 - j] = a[j];

}

//print the reversed array using a for loop

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

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

}

//print a new line

System.out.println();

//if original and reversed arrays are not equal, return false

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

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

return false;

}

}

//if arrays are equal, return true

return true;

}

}

**Program 2 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the length of the array
4. Declare an int array variable of the given length to store numbers from the user
5. Prompt the user to enter values for the array
6. Add the user’s values into the array using a for loop
7. Create a separate method to find the second largest number in the array – this method should take in an int array and return an int
8. Sort the array
9. Use a decrementing for loop to find the second largest number – should start at the index of length-2 as the max will be at the index length-1 after the array is sorted
10. If the current array number is not equal to the last number in the array, it is the second largest number
11. Return the second largest number
12. If the for loop does not return anything, return -1 (just put at end of method)
13. End the method to find the second largest number and return to the main method
14. Call the method that determines the second largest number using the original array as a parameter – if it returns -1, print an error message – otherwise, print the second largest number
15. Create a separate method to find the third smallest number in the array – this method should take in an int array and return an int
16. Sort the array
17. Declare int variables for the position of the second smallest number and the value of the third smallest number
18. Use an incrementing for loop to find the position of the second smallest number (similar to process in second largest method, just going the other way)
19. If the current array number is not equal to the first number in the array (the minimum), it is the second smallest number
20. Update the position of the second smallest number and break out of the for loop
21. Use another incrementing for loop to find the third smallest number – should start at the index of the second smallest number+1
22. If the current array number is not equal to the second smallest number, it is the third smallest number
23. Return the third smallest number
24. If the for loop does not return anything, return -1 (just put at end of method)
25. End the method to find the third smallest number and return to the main method
26. Call the method that determines the third smallest number using the original array as a parameter – if it returns -1, print an error message – otherwise, print the third smallest number
27. End the main method
28. End the program

**Program 2 Running Screenshot**

**Text

Description automatically generated**

**Program 2 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P2SecondLargestAndThirdSmallest: int, int array ; int int (possibly string)

program takes in a number from the user for the length of the array as well as numbers to

fill that array then determines and prints out the second largest number in the array and

the third smallest number in the array

ex1: user inputs 8, 4, 6, 3, 5, 5, 3, 6, 4 - program outputs 5, 5

ex2: user inputs 4, 1, 2, 2, 1 - program outputs 1, Error calculating third smallest number. -- this happens because there is no third smallest number

ex3: user inputs 11, 8, 7, 4, 7, 6, 5, 3, 2, 4, 6, 5 - program outputs 7, 4

ex4: user inputs 7, -5, 54, 87, 10, 11, 10, 87 - program outputs 54, 11

ex5: user inputs 5, 1.3, x, y, x, 1.3 - program outputs error

ex6: user inputs 2.0, 12, 12 - program outputs error

\*/

import java.util.Arrays;

import java.util.Scanner;

public class P2SecondLargestAndThirdSmallest {

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 length = input.nextInt();

//declare an int array variable of the input length to store the numbers input by the user

int[] userNumbers = new int[length];

//prompt the user to enter values for the array

System.out.print("Enter values to be put in array: ");

//add the user's values into the array using a for loop

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

userNumbers[i] = input.nextInt();

}

//call the method that determines the second largest number using the original array as a parameter - if it returns -1, give an error message, otherwise print second largest number

if(secondLargest(userNumbers) == -1) {

System.out.println("Error calculating second largest number.");

}

else{

System.out.println("Second largest = " + secondLargest(userNumbers));

}

//call the method that determines the third smallest number using the original array as a parameter - if it returns -1, give an error message, otherwise print the third smallest number

if(thirdSmallest(userNumbers) == -1){

System.out.println("Error calculating third smallest number.");

}

else{

System.out.println("Third smallest = " + thirdSmallest(userNumbers));

}

}

//create a separate method to find the second largest number - should take in an int array and return an int

public static int secondLargest(int[] a){

//sort the array

Arrays.sort(a);

//declare an int variable for the second largest number

int secondLargest;

//use a decrementing for loop to find the second largest number - start at length - 2 as max should be at index length - 1

for(int j = a.length - 2; j >= 0; j--){

//if the number is not equal to the last number in the array, it is the second largest number

if(a[j] != a[a.length - 1]){

secondLargest = a[j];

//return the second largest number

return secondLargest;

}

}

//if for loop does not return value, return -1

return -1;

}

//create a separate method to find the third smallest number - should take in an int array and return an int

public static int thirdSmallest(int[] b){

//sort the array

Arrays.sort(b);

//declare an int variable for the position of the second smallest number and the value of the third smallest number

int positionSecondSmallest = b.length - 1;

int thirdSmallest;

//use an incrementing for loop to find the position of the second smallest number

for(int k = 1; k < b.length; k++){

//if the number is not equal to the first number in the array, it is the second smallest number

if(b[k] != b[0]){

positionSecondSmallest = k;

break;

}

}

//use another incrementing for loop starting at the position of the second smallest number + 1 to find the third smallest number

for(int x = positionSecondSmallest + 1; x < b.length; x++){

//if the number is not equal to the second smallest number, it is the third smallest number

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

thirdSmallest = b[x];

//return the third smallest number

return thirdSmallest;

}

}

//if the for loop does not return anything, return -1

return -1;

}

}

**Program 3 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the desired sum and declare an int variable for it
4. Prompt the user to enter the length of the number array and declare an int variable for it
5. Declare an int array variable of the given length
6. Prompt the user to enter values for the array of numbers
7. Use a for loop to enter the user’s input into the array of numbers
8. Declare an int variable for the number of pairs that add up to the desired sum
9. Use nested for loops to find how many pairs add to the desired sum
10. If two numbers at different spots in the array add to the desired sum, add 1 to the number of pairs that add up to the sum
11. Print the pairs that add to the sum
12. Print a new line
13. Print the number of pairs that add up to the desired sum
14. End the program

**Program 3 Running Screenshot**

**Text

Description automatically generated**

**Program 3 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P3PairsWithSum: int, int, int array ; int

program takes in a number from the user for the desired sum, a number for the length of the array,

and numbers to fill that array - then determines and prints out how many pairs of numbers give that

sum (also prints the pairs)

ex1: user inputs 6, 4, 1 , 5, 7, -1 - program outputs 2

ex2: user inputs 6, 5, 1, 5, 7, -1, 5 - program outputs 3

ex3: user inputs 2, 4, 1, 1, 1, 1 - program outputs 6

ex4: user inputs 11, 13, 10, 12, 10, 15, -1, 7, 6, 5, 4, 2, 1, 1, 1 - program outputs 9

ex5: user inputs 5, 4, 1.3, x, y, x - program outputs error

ex6: user inputs 2.0, 2, 12, 12 - program outputs error

\*/

import java.util.Scanner;

public class P3PairsWithSum {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt the user to enter the desired sum and declare an int variable for it

System.out.print("Enter the desired sum: ");

int sum = input.nextInt();

//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 length = input.nextInt();

//declare an int array variable of the given length for the array of numbers

int[] numArray = new int[length];

//prompt the user to enter values for the array

System.out.print("Enter values to be put in the array: ");

//use a for loop to enter the user's values into the array

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

numArray[i] = input.nextInt();

}

//declare an int variable for the number of pairs that add to the desired sum

int numPairs = 0;

System.out.print("Pairs that add to " + sum + ": ");

//use nested for loops to find how many pairs of numbers add to the desired sum

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

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

//if two of the numbers in the array add to the desired sum, add 1 to the number of pairs

if(numArray[j] + numArray[k] == sum){

numPairs++;

//print the pair that add to the sum

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

}

}

}

//print a new line

System.out.println();

//print the number of pairs that add to the sum

System.out.println(numPairs + " pairs add to " + sum);

}

}

**Program 4 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the value to make change for and declare an int variable for this value
4. Prompt the user to enter the number of coins (length of the array) and declare an int variable for this value
5. Declare an int array variable of the length given by the user
6. Prompt the user to enter the values of the coins (numbers in the array)
7. Use a for loop to enter the user’s input into the array
8. Create a separate recursive method to determine the number of ways to make change – should take in two ints (desired change and array length) and an int array (coin values) and it should return an int
9. Check to see if the user wants to make change for 0 – if so, return 1
10. Check to see if the user wants to make change for a negative number – if so, return 0
11. Check to see if there are any coins in the array to make change with (check if array length is greater than 0 or not) – if not, return 0
12. Declare an int variable for the number of ways to make change
13. Use this recursive method to go through each element of the array and determine the number of ways to make change for the desired value
14. Return the number of ways to make change
15. End the method that calculates the number of ways to make change and return to the main method
16. Call the method that determines the number of ways change can be made in a print statement – use the desired change value, number of coins (array length), and the array containing the coin values as parameters
17. End the main method
18. End the program

**Program 4 Running Screenshot**

**Text

Description automatically generated**

**Program 4 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P4CoinChange: int, int, int array ; int

program takes in a number from the user for the value to make change for, a number for the length of the array,

and coin values to fill that array - program then determines and prints out how many ways the given coin values

can make change for the value given by the user

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

ex2: user inputs 10, 4, 2, 5, 3, 6 - program outputs 5

ex3: user inputs 11, 5, 2, 4, 5, 9, 13 - program outputs 3

ex4: user inputs 15, 6, 10, 12, 2, 6, 8, 20 - program outputs 0

ex5: user inputs 151.2, 2, 1, 2 - program outputs error

ex6: user inputs 10, 3, 1, 2, 3.5 - program outputs error

\*/

import java.util.Scanner;

public class P4CoinChange {

public static void main(String[] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt the user to enter the value to make change for and declare an int variable for it

System.out.print("Enter the number of cents you want to make change for: ");

int totalChange = input.nextInt();

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

System.out.print("Enter the number of coins you want to use to make change: ");

int numOfCoins = input.nextInt();

//declare an int array variable of the length given by the user

int[] coins = new int[numOfCoins];

//prompt the user to enter the values of the coins (numbers in the array)

System.out.print("Enter the values of the coins you want to use to make change: ");

//use a for loop to put the user's values into the array

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

coins[i] = input.nextInt();

}

//call the method that determines the number of ways to make change in a print statement - use the desired change value, the number of coins (length of array), and the array with the value of the coins as parameters

System.out.println(makeChange(totalChange, numOfCoins, coins) + " ways to make change for " + totalChange + " cents");

}

//create a separate (and recursive) method to determine the number of ways change can be made - should take in 2 ints (number to make change for and array length) and an int array (value of coins) and return an int

public static int makeChange(int a, int b, int[] c){

//check to see if user wants to make change for 0 - if so, return 1

if(a == 0){

return 1;

}

//check to see if user wants to make change for a negative number - if so, return 0

if(a < 0){

return 0;

}

//check to see if there are coins to make change with - if not, return 0

if(b <= 0){

return 0;

}

//declare an int variable for the number of ways to make change

int waysToMakeChange = 0;

//use this recursive method to find the number of ways to make change and update the number of ways to make change

waysToMakeChange = makeChange(a, b - 1, c) + makeChange(a - c[b-1], b, c);

//return the number of ways change can be made

return waysToMakeChange;

}

}

**Program 5 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the number of mangoes and declare an int variable for it
4. Prompt the user to enter the number of people and declare an int variable for it
5. Declare an int variable for the factorial of the number of mangoes and use a for loop to calculate this value – will be used later on
6. Declare an int variable for the factorial of the number of people – 1 and use a for loop to calculate this value – will be used later on as well
7. Create a separate method to calculate the binomial coefficient – should take in two ints and return an int
8. Declare an int variable for the coefficient
9. Use a for loop to calculate the coefficient
10. Return the coefficient
11. End the method that determines the binomial coefficient
12. Create a separate method to calculate the number of ways mangoes can be distributed for case 1 – should take in two ints and return an int
13. Check if mangoes are less than people – if so, return 0
14. Declare an int variable for the number of ways the mangoes can be distributed in case 1 – calculate value by calling method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters
15. Return the number of ways mangoes can be distributed in case 1
16. End the method for case 1
17. Create a separate method to calculate the number of ways mangoes can be distributed for case 2 – should take in three ints and return an int
18. Check if mangoes are less than people – if so, return 0
19. Declare an int variable for the number of ways the mangoes can be distributed in case 2 – calculate value by calling method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters
20. Multiply the number of ways mangoes can be distributed in case 2 by the factorial of the number of mangoes
21. Return the number of ways mangoes can be distributed in case 2
22. End the method for case 2
23. Create a separate method to calculate the number of ways mangoes can be distributed for case 3 – should take in three ints and return an int
24. Check if mangoes are less than people – if so, return 0
25. Declare an int variable for the number of ways the mangoes can be distributed in case 3 – calculate value by calling method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters
26. Multiply the number of ways mangoes can be distributed in case 3 by the factorial of the number of people - 1
27. Return the number of ways mangoes can be distributed in case 3
28. End the method for case 3
29. Create a separate method to calculate the number of ways mangoes can be distributed for case 4 – should take in 4 ints and return an int
30. Check if mangoes are less than people – if so, return 0
31. Declare an int variable for the number of ways the mangoes can be distributed in case 4 – calculate value by calling method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters
32. Multiply the number of ways mangoes can be distributed in case 4 by the factorial of the number of mangoes and by the factorial of the number of people - 1
33. Return the number of ways mangoes can be distributed in case 4
34. End the method for case 4 and return to the main method
35. In a print statement, call the method that determines the number of ways to distribute the mangoes for case 1 using mangoes and people as parameters
36. In a print statement, call the method that determines the number of ways to distribute the mangoes for case 2 using mangoes, people, and factorial of mangoes as parameters
37. In a print statement, call the method that determines the number of ways to distribute the mangoes for case 3 using mangoes, people, and factorial of people - 1 as parameters
38. In a print statement, call the method that determines the number of ways to distribute the mangoes for case 4 using mangoes, people, factorial of mangoes, and factorial of people - 1 as parameters
39. End the main method
40. End the program

**Program 5 Running Screenshot**

**Graphical user interface, text

Description automatically generated**

**Program 5 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P5DistributeMangoesToPeople: int, int ; int, int, int, int

program takes in a number from the user for number of mangoes and a number for the number

of people and calculates and prints the number of ways the mangoes can be distributed for

each of the 4 cases below

4 cases:

1. mangoes and people considered identical -- number of ways = (m+n-1)C(n-1)

2. mangoes unique, people identical -- number of ways = (m+n-1)C(n-1) \* m!

3. mangoes identical, people unique -- number of ways = (m+n-1)C(n-1) \* (n-1)!

4. mangoes and people considered unique -- number of ways = (m+n-1)C(n-1) \* (n-1)! \* m!

ex1: user inputs 3, 2 - program outputs 4, 24, 4, 24

ex2: user inputs 7, 5 - program outputs 330, 1663200, 7920, 39916800

ex3: user inputs 9, 6 - program outputs 2002, 726485760, 240240, 1278945280

ex4: user inputs 2, 3 - program outputs 0, 0, 0, 0 -- can't properly distribute mangoes when there are more people than mangoes

ex5: user inputs 2.5, 3 - program outputs error

ex6: user inputs x, z - program outputs error

\*/

import java.util.Scanner;

public class P5DistributeMangoesToPeople {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt user to enter number of mangoes and declare an int variable for this value

System.out.print("Enter the number of mangoes to be distributed: ");

int mangoes = input.nextInt();

//prompt user to enter number of people and declare an int variable for this value

System.out.print("Enter the number of people to distribute mangoes to: ");

int people = input.nextInt();

//declare an int variable for the factorial of the number of mangoes

int mangoFactorial = 1;

//use a for loop to calculate the factorial of the number of mangoes

for (int i = 1; i <= mangoes; i++){

mangoFactorial = mangoFactorial \* i;

}

//declare an int variable for the factorial of the number of people - 1

int peopleFactorial = 1;

//use a for loop to calculate the factorial of the number of people - 1

for (int j = 1; j <= people - 1; j++){

peopleFactorial = peopleFactorial \* j;

}

//in a print statement, call the method that determines number of ways for case 1 using number of mangoes and people as parameters

System.out.println("Case 1: " + waysToDistributeCase1(mangoes, people) + " ways");

//in a print statement, call the method that determines number of ways for case 2 using number of mangoes, people, and factorial of mangoes as parameters

System.out.println("Case 2: " + waysToDistributeCase2(mangoes, people, mangoFactorial) + " ways");

//in a print statement, call the method that determines number of ways for case 3 using number of mangoes, people, and factorial of people - 1 as parameters

System.out.println("Case 3: " + waysToDistributeCase3(mangoes, people, peopleFactorial) + " ways");

//in a print statement, call the method that determines number of ways for case 4 using number of mangoes, people, factorial of mangoes, and factorial of people - 1 as parameters

System.out.println("Case 4: " + waysToDistributeCase4(mangoes, people, mangoFactorial, peopleFactorial) + " ways");

}

//create a separate method to calculate the binomial coefficient - should take in two ints and return an int

public static int binomCoefficient (int x, int y){

//declare an int variable for the coefficient

int coefficient = 1;

if(y > x - y){

y = x - y;

}

//use a for loop to calculate the coefficient

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

coefficient = coefficient \* (x - i);

coefficient = coefficient / (i + 1);

}

//return the coefficient

return coefficient;

}

//create a function that calculates the number of ways mangoes can be distributed for case 1 - should take in two ints and return an int

public static int waysToDistributeCase1 (int a, int b){

//check if mangoes are less than people - if so, return 0

if(a < b){

return 0;

}

//declare an int variable for the number of ways the mangoes can be distributed

//calculate total ways by calling the method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters

int waysToDistribute1 = binomCoefficient(a + b - 1, b - 1);

//return the number of ways mangoes can be distributed

return waysToDistribute1;

}

//create a function that calculates the number of ways mangoes can be distributed for case 2 - should take in three ints and return an int

public static int waysToDistributeCase2 (int c, int d, int e){

//check if mangoes are less than people - if so, return 0

if(c < d){

return 0;

}

//declare an int variable for the number of ways the mangoes can be distributed

//calculate total ways by calling the method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters

int waysToDistribute2 = binomCoefficient(c + d - 1, d - 1);

//multiply the ways mangoes can be distributed by the factorial of the number of mangoes (e)

waysToDistribute2 = waysToDistribute2 \* e;

//return the number of ways mangoes can be distributed

return waysToDistribute2;

}

//create a function that calculates the number of ways mangoes can be distributed for case 3 - should take in three ints and return an int

public static int waysToDistributeCase3 (int f, int g, int h){

//check if mangoes are less than people - if so, return 0

if(f < g){

return 0;

}

//declare an int variable for the number of ways the mangoes can be distributed

//calculate total ways by calling the method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters

int waysToDistribute3 = binomCoefficient(f + g - 1, g - 1);

//multiply the ways mangoes can be distributed by the factorial of the number of people - 1 (h)

waysToDistribute3 = waysToDistribute3 \* h;

//return the number of ways mangoes can be distributed

return waysToDistribute3;

}

//create a function that calculates the number of ways mangoes can be distributed for case 4 - should take in 4 ints and return an int

public static int waysToDistributeCase4 (int r, int s, int t, int u){

//check if mangoes are less than people - if so, return 0

if(r < s){

return 0;

}

//declare an int variable for the number of ways the mangoes can be distributed

//calculate total ways by calling the method that calculates the binomial coefficient using mangoes + people - 1 and people - 1 as parameters

int waysToDistribute4 = binomCoefficient(r + s - 1, s - 1);

//multiply the ways mangoes can be distributed by the factorial of the number of mangoes (t) and by the factorial of people - 1 (u)

waysToDistribute4 = waysToDistribute4 \* t \* u;

//return the number of ways mangoes can be distributed

return waysToDistribute4;

}

}

**Program 6 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the dimensions of the first matrix and declare int variables for the number of rows and columns in the first matrix
4. Declare a 2D int array with the dimensions provided by the user
5. Prompt the user to enter values for the first matrix
6. Use a nested for loop to enter the values into the first matrix
7. Prompt the user to enter the dimensions of the second matrix and declare int variables for the number of rows and columns in the second matrix
8. Declare a 2D int array with the dimensions provided by the user
9. Prompt the user to enter values for the first matrix
10. Use a nested for loop to enter the values into the second matrix
11. Create a separate method that multiplies two matrices – should use the dimensions of array 1 and array 2 as well as the two 2D arrays as parameters (so, 4 int variables and 2 int[][] array variables as parameters)
12. Check if the columns of the first matrix and the rows of the second matrix are equal – if not, print a message telling the user multiplication is not possible and exit the method
13. Declare a new 2D array to store the product matrix – should have the number of rows in the first matrix and the number of columns in the second matrix
14. Use nested for loops to calculate the product matrix
15. Use another nested for loop to print the product matrix – print each row then print a new line and go to the next row
16. End the method that multiplies matrices and return to the main method
17. Call the method that multiplies two matrices using the dimensions of array 1 and array 2 and the two arrays as parameters
18. End the program

**Program 6 Running Screenshot**

**Text

Description automatically generated**

**Program 6 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P6MultiplyMatrices: int, int, int, int, int array, int array ; number of ints in the matrix (in the correct matrix format)

program takes in numbers from the user for the number of rows and columns in the first array,

number of rows and columns in the second array, values for the first and second arrays, then

calculates and prints the product of the two matrices

ex1: user inputs 2, 2, 2, 4, 3, 4, 2, 2, 1, 2, 1, 3 - program outputs 6, 16, 7, 18

ex2: user inputs 2, 2, 1, 2, 3, 4, 2, 2, 1, 1, 1, 1 - program outputs 3, 3, 7, 7

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

ex4: user inputs 2, 2, 1, 2, 3, 4, 1, 2, 11, 12 - program outputs Multiplication is not possible with given dimensions

ex5: user inputs 2.1, 1, 1.2, 2, 1, 1, 0.5 - program outputs error

ex6: user inputs 2, 1, x, y, 1, 2, 3, 4 - program outputs error

\*/

import java.util.Scanner;

public class P6MultiplyMatrices {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt the user to enter the rows and columns in the first array and declare int variables for these values

System.out.print("Enter the number of rows in the first array followed by the number of columns: ");

int rowsArray1 = input.nextInt();

int columnsArray1 = input.nextInt();

//declare a 2D int array with the first set of dimensions

int[][] array1 = new int[rowsArray1][columnsArray1];

//prompt the user to enter values for the first matrix

System.out.print("Enter values for the first matrix: ");

//use nested for loop to enter values into first array

for(int i = 0; i < rowsArray1; i++){

for(int j = 0; j < columnsArray1; j++){

array1[i][j] = input.nextInt();

}

}

//prompt the user to enter the rows and columns in the second array and declare int variables for these values

System.out.print("Enter the number of rows in the first array followed by the number of columns: ");

int rowsArray2 = input.nextInt();

int columnsArray2 = input.nextInt();

//declare a 2D int array with the second set of dimensions

int[][] array2 = new int[rowsArray2][columnsArray2];

//prompt the user to enter values for the second matrix

System.out.print("Enter values for the second matrix: ");

//use nested for loop to enter values into second array

for(int i = 0; i < rowsArray2; i++){

for(int j = 0; j < columnsArray2; j++){

array2[i][j] = input.nextInt();

}

}

//call the method that multiplies two matrices using the dimensions of array 1 and array 2 and the two arrays as parameters

multiplyMatrices(rowsArray1, columnsArray1, rowsArray2, columnsArray2, array1, array2);

}

//create a separate method that multiplies two matrices - should take in 4 ints (the dimensions of each array) and 2 2D int arrays and return nothing

public static void multiplyMatrices (int a, int b, int c, int d, int[][] x, int[][] y){

//check if the columns for array1 (b) and rows for array2 (c) are equal - if not, tell user multiplication is not possible and leave method

if(b != c){

System.out.println("Multiplication is not possible with given dimensions");

return;

}

//declare a new 2D array to store the product matrix - should have number of rows in array 1 (a) and columns in array 2 (d)

int[][] product = new int[a][d];

//use nested for loops to calculate the product matrix

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

for(int j = 0; j < d; j++){

for(int k = 0; k < c; k++){

product[i][j] += x[i][k] \* y[k][j];

}

}

}

//use a nested for loop to print the product matrix - print each row then print a new line and go to the next row

for(int m = 0; m < a; m++){

for(int n = 0; n < d; n++){

System.out.print(product[m][n] + " ");

}

System.out.println();

}

}

}

**Program 7 Algorithm**

1. Start the program
2. Import the scanner
3. Prompt the user to enter the dimensions of the first matrix and declare int variables for the number of rows and columns in the first matrix
4. Declare a 2D int array with the dimensions provided by the user
5. Prompt the user to enter values for the first matrix
6. Use a nested for loop to enter the values into the first matrix
7. Prompt the user to enter the dimensions of the second matrix and declare int variables for the number of rows and columns in the second matrix
8. Declare a 2D int array with the dimensions provided by the user
9. Prompt the user to enter values for the first matrix
10. Use a nested for loop to enter the values into the second matrix
11. Create a separate method that adds two matrices – should use the dimensions of array 1 and array 2 as well as the two 2D arrays as parameters (4 int variables total and 2 int[][] array variables)
12. Check if the dimensions of the first matrix and second matrix are equal – if not, print a message telling the user addition is not possible and exit the method
13. Declare a new 2D array to store the sum matrix – should have the same dimensions as both of the original arrays
14. Use nested for loops to calculate the sum matrix
15. Use another nested for loop to print the sum matrix – print contents in each row then print a new line and go to the next row
16. End the method that adds matrices and return to the main method
17. Call the method that adds two matrices using the dimensions of array 1 and array 2 and the two arrays as parameters
18. End the program

**Program 7 Running Screenshot**

**Text

Description automatically generated**

**Program 7 Code**

/\*

Christopher Bussen

CPS 150 02

Assignment 6

P7AddMatrices: int, int, int, int, int array, int array ; number of ints in the matrix (in the correct matrix format)

program takes in numbers from the user for the number of rows and columns in the first array,

number of rows and columns in the second array, values for the first and second arrays, then

calculates and prints the sum of the two matrices

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

ex2: user inputs 2, 2, 11, 9, -7, 102, 2, 2, -4, -19, 22, 4 - program outputs 7, -10, 15, 106

ex3: user inputs 1, 2, 3, 6, 2, 1, 7, -4 - program outputs Addition is not possible with given dimensions

ex4: user inputs 2, 2, 1, 2, 3, 4, 1, 2, 11, 12 - program outputs Addition is not possible with given dimensions

ex5: user inputs 2.1, 1, 1.2, 2, 1, 1, 0.5 - program outputs error

ex6: user inputs 2, 1, x, y, 1, 2, 3, 4 - program outputs error

\*/

import java.util.Scanner;

public class P7AddMatrices {

public static void main(String [] args){

//import scanner

Scanner input = new Scanner(System.in);

//prompt the user to enter the rows and columns in the first array and declare int variables for these values

System.out.print("Enter the number of rows in the first array followed by the number of columns: ");

int rowsArray1 = input.nextInt();

int columnsArray1 = input.nextInt();

//declare a 2D int array with the first set of dimensions

int[][] array1 = new int[rowsArray1][columnsArray1];

//prompt the user to enter values for the first matrix

System.out.print("Enter values for the first matrix: ");

//use nested for loop to enter values into first array

for(int i = 0; i < rowsArray1; i++){

for(int j = 0; j < columnsArray1; j++){

array1[i][j] = input.nextInt();

}

}

//prompt the user to enter the rows and columns in the second array and declare int variables for these values

System.out.print("Enter the number of rows in the first array followed by the number of columns: ");

int rowsArray2 = input.nextInt();

int columnsArray2 = input.nextInt();

//declare a 2D int array with the second set of dimensions

int[][] array2 = new int[rowsArray2][columnsArray2];

//prompt the user to enter values for the second matrix

System.out.print("Enter values for the second matrix: ");

//use nested for loop to enter values into second array

for(int i = 0; i < rowsArray2; i++){

for(int j = 0; j < columnsArray2; j++){

array2[i][j] = input.nextInt();

}

}

//call the method that multiplies two matrices using the dimensions of array 1 and array 2 and the two arrays as parameters

addMatrices(rowsArray1, columnsArray1, rowsArray2, columnsArray2, array1, array2);

}

//create a separate method that adds two matrices - should take in 4 ints (the dimensions of each array) and 2 2D int arrays and return nothing

public static void addMatrices (int a, int b, int c, int d, int[][] x, int[][] y) {

//check if the dimensions for array1 (a and b) and array2 (c and d) are equal - if not, tell user addition is not possible and leave method

if (a != c || b != d) {

System.out.println("Addition is not possible with given dimensions");

return;

}

//declare a new 2D array to store the sum matrix - should have same dimensions as array 1 and array 2

int[][] sum = new int[a][b];

//use nested for loops to calculate the sum matrix

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

for (int j = 0; j < b; j++) {

sum[i][j] = x[i][j] + y[i][j];

}

}

//use a nested for loop to print the sum matrix - print each row then print a new line and go to the next row

for(int k = 0; k < a; k++){

for(int z = 0; z < d; z++){

System.out.print(sum[k][z] + " ");

}

System.out.println();

}

}

}