CSC1003 Assignment 3

OJ Access Code: gjw&$9d5pl

## Important Notes:

1. The assignment is an individual project, to be finished on one’s own effort.
2. The work must be submitted before 6pm Dec. 8, 2023 (Friday), Beijing Time. This is a firm deadline. No late submissions are accepted.
3. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% marks will be deducted.

## Marking Criterion:

1. There are three java programs in this assignment, 45 marks each.
2. The maximum score of assignments 1 and 2 is 100 marks each. The maximum score of assignments 3 and 4 is 135 marks each. Meanwhile, the maximum total score of all four assignments for each student can’t exceed 400.
3. Each program will be evaluated with several unseen test cases. A program obtains the full score if and only if it passes all test cases.
4. According to the school policy, using AI will be regarded as academic dishonesty. If TA suspects that your code was generated by AI, you will be invited to an offline meeting and express your understanding of the assignment. Otherwise, you may receive a minimum grade.

## Running Environment:

1. The submissions will be evaluated in the course’s OJ system running Java SE version 17 and Linux platform.
2. The submission is only allowed to import four packages of (java.lang.\*; java.util.\*; java.math.\*; java.io.\*) included in Java SDK. No other packages are allowed.
3. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.

## Submission Guidelines:

1. You will receive your grade only if you submit your code both on OJ and on bb on time. Latesubmission on bb leads to 10% mark deduction, and late submission on OJ leads to minimumgrade.
2. For bb submission, you need to directly upload your java file on bb. That is, your submission should be “RegionFill.java”, “ClassVector.java”, and “TestMathExpr.java”. Wrong submission format will receive 10% mark deduction.
3. Inconsistency with or violation from the guideline leads to marks deduction.
4. It is the students’ responsibility to read this assignment document and submission guidelines carefully and in detail. No argument will be accepted on issues that have been specified clearly in these documents.

## Program I:

Write a Java program (RegionFill.java) to simulate the process of filling color to a closed region. The region is formed by a closed curve and/or the boundary. The program starts from a point with given coordinates and fill in the “color” for all pixels inside the closed region.

|  |  |
| --- | --- |
| Example input | Expected output |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*-----\*\*\*\*\*\*\*\*\*\*\*\*\*  \*-\*\*\*\*\*-\*\*\*\*\*\*\*\*\*\*\*\*  \*-\*\*\*\*\*\*-\*\*\*\*\*\*\*\*\*\*\*  \*\*-\*\*\*\*\*\*-\*\*\*\*\*\*\*\*\*\*  \*\*\*-\*\*\*\*\*\*-\*\*\*\*\*\*\*\*\*  \*\*\*-\*\*\*\*\*\*\*-\*\*\*\*\*\*\*\*  \*\*\*-\*\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*-\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*-\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*-\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*\*-\*\*\*\*\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*-\*\*\*\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*--\*\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*-\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  9, 11 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*-----\*\*\*\*\*\*\*\*\*\*\*\*\*  \*-------\*\*\*\*\*\*\*\*\*\*\*\*  \*--------\*\*\*\*\*\*\*\*\*\*\*  \*\*--------\*\*\*\*\*\*\*\*\*\*  \*\*\*--------\*\*\*\*\*\*\*\*\*  \*\*\*---------\*\*\*\*\*\*\*\*  \*\*\*----------\*\*\*\*\*\*\*  \*\*\*\*---------\*\*\*\*\*\*\*  \*\*\*\*\*--------\*\*\*\*\*\*\*  \*\*\*\*\*\*-------\*\*\*\*\*\*\*  \*\*\*\*\*\*\*-------\*\*\*\*\*\*  \*\*\*\*\*\*\*\*------\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*-----\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*---\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*-----\*\*\*\*\*\*\*\*\*\*\*\*\*  \*-\*\*\*\*\*-\*\*\*\*\*\*\*\*\*\*\*\*  \*-\*\*\*\*\*\*-\*\*\*\*\*\*\*\*\*\*\*  \*\*-\*\*\*\*\*\*-\*\*\*\*\*\*\*\*\*\*  \*\*\*-\*\*\*\*\*\*-\*\*\*\*\*\*\*\*\*  \*\*\*-\*\*\*\*\*\*\*-\*\*\*\*\*\*\*\*  \*\*\*-\*\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*-\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*-\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*-\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*\*-\*\*\*\*\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*-\*\*\*\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*--\*\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*-\*-\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*-\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  18, 0 | --------------------  --------------------  --\*\*\*\*\*-------------  --\*\*\*\*\*\*------------  ---\*\*\*\*\*\*-----------  ----\*\*\*\*\*\*----------  ----\*\*\*\*\*\*\*---------  ----\*\*\*\*\*\*\*\*--------  -----\*\*\*\*\*\*\*--------  ------\*\*\*\*\*\*--------  -------\*\*\*\*\*--------  --------\*\*\*\*\*-------  ---------\*\*\*\*-------  -----------\*\*-------  ------------\*-------  --------------------  --------------------  --------------------  --------------------  -------------------- |

The console input is given by 20 \* 20 characters of either “\*” or “-“. A closed region can be formed by the “-“ characters, or by the “-“ characters and the “\*” characters in the boundary. Following the input of the characters, two numbers of the (x, y) coordinates of the starting point are given, separated by a comma and a space.

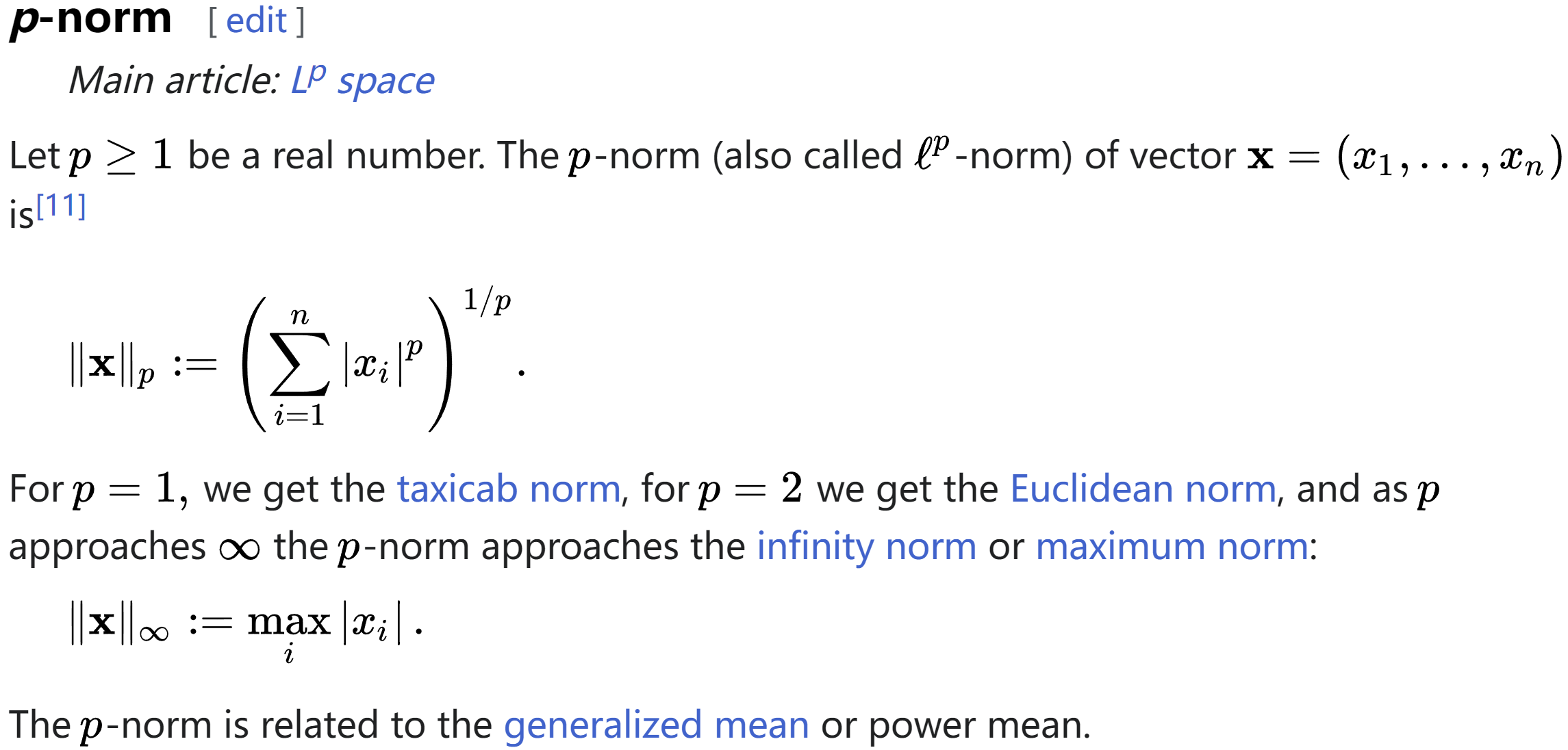
In the first example above, the starting point is (9, 11), which is the 12-th character of line 10. The corresponding output of the result is shown in the right.

In the second example above, the starting point is (18, 0), which is the 1st character of line 19. For this starting point, the corresponding output of the result is shown in the right.

## Program II:

Write a Java program (ClassVector.java) according to the following tutorial:

1. Create a class “Vector” which is used to store an n-dimensional vector and do relevant operations. Write a constructor for “Vector” that takes an int array as parameter.
2. Write a static method “plus(Vector v1, Vector v2)” that returns the sum of v1 and v2, which is also a Vector. Also, write a static method “subtract(Vector v1, Vector v2)” that returns v1 – v2.
3. Write a non-static method “multiply(int a)” that returns the result of multiplying a Vector by the integer a.
4. Write a non-static method “norm(int p)” that returns the p-norm of a Vector as a double. The definition of norm is given below:



(<https://en.wikipedia.org/wiki/Norm_(mathematics)#p-norm>)

Overload the method so that it will return the infinity-norm of the Vector when taking no parameter.

1. Write a static method “dot(Vector v1, Vector v2)” that returns the dot product of v1 and v2 as an integer.
2. Write a static method “angle(Vector v1, Vector v2)” that returns the angle between v1 and v2 as a double, provided that v1 and v2 are not 0 vector.
3. Override the non-static method “toString()” that returns a String in the form:

(v1, v2, …, vn)

|  |  |
| --- | --- |
| Example input | Expected output |
| 2  2 3 5 6  0 5 6 8  plus 0 1  subtract 1 0  multiply 1 -3  norm 1 1  norm 0 infinity  dot 0 1  angle 0 1 | (2, 8, 11, 14)  (-2, 2, 1, 2)  (0, -15, -18, -24)  19.000  6.000  93  0.258 |

Note:

1. Please download the template from bb.
2. You only need to implement the class Vector. That is, you do NOT need to deal with input and output by yourself.
3. Do **NOT** modify the “main” part, which is used to test your code.
4. You may need java.lang.Math:

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html>

## Program Three:

Write a Java Program (TestMathExpr.java) with the following requirement.

1. It evaluates the value of math expressions, and outputs an integer value.
2. Each math expression includes (see the example below):
   1. numbers (integers and doubles);
   2. (no more than five) operators of “+” (addition), “-“ (subtraction), “\*” (multiplication) and “/” (division);
   3. (no more than three) functions including “sin” (sine function), “cos” (cosine function), “tan” (tangent function) and “sqrt” (square root function).
   4. “(“ and “)” (brackets);
   5. possibly blank space.
3. All expressions are valid. The output is an integer value after rounding.

|  |  |
| --- | --- |
| Example input | Expected output |
| 1+2.0\*sin(37+(25\*3))  (2+ 3.50)\*4\*sqrt(sin(1.5))  -3+4/ (2.5+3.7)  (-3+4)/2.5+3.9  1.2-3.5\*5.2-13.2  1.2-3.5\*5.2-13.7  2.3\*5\*7 - 12\*9/8  -sin(3.5-sqrt(4)) + cos(tan(2.5)) | -1  22  -2  4  -30  -31  67  0 |

**Note:** Each submission is expected to strictly follow the following template to implement the required function by modifying the **parse()** function/method.

Text

Description automatically generated