This document is in compliance with SMDS standard version 4.2

|  |  |
| --- | --- |
| **Description:** | Determines whether an image is a rotation, reflection, a combination of the two, or is none of the above. |

**Development Estimates/Actuals**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Item** | **Est Sz** | **Act Sz** |  | **Strt Date** | **Est Cmplt** | **Act Cmplt** |  | **Est Effrt** | **Act Effrt** |
| Reqs | 10 |  |  | 11/24 | 11/24 | 11/24 |  | 1 hr | 1 hr |
| Inpt Partition |  |  |  | 11/24 | 11/24 | 11/24 |  | 2 hrs | 1.5 hrs |
| Test Cases |  |  |  | 11/25 | 11/25 | 11/25 |  | 1 hr | 1.5 hrs |
| Design |  |  |  | 11/30 | 11/30 | 11/30 |  | 3 hrs | 2.5 hrs |
| # Classes |  |  |  |  |  |  |  |  |  |
| # Methods |  |  |  |  |  |  |  |  |  |
| # Dsng Elms |  |  |  |  |  |  |  |  |  |
| Algor Correct |  |  |  | 11/30 | 11/30 | 11/30 |  | 1.5 hrs | 0.5 hrs |
| Implementatn |  |  |  | 12/1 | 12/1 | 12/2 |  | 5 hrs | 3 hrs |
| NCLC |  |  |  |  |  |  |  |  |  |
| Requir. Trace |  |  |  |  |  |  |  |  |  |
| Code Correct |  |  |  | 12/11 | 12/11 | 12/11 |  | 2 hrs | 1.5 hrs |
| Final Test |  |  |  | 12/11 | 12/11 | 12/11 |  | 1.5 hrs | 1 hr |
| Inspec |  |  |  |  |  |  |  | 0 hrs | 0 hrs |
| Wrap up |  |  |  | 12/12 | 12/12 | 12/12 |  | 1 hr | 2 hrs |
|  |  |  |  |  |  | Total Hrs |  | 18 hrs | 14.5 hrs |

Estimates/Actuals Comments

* Comments and time log can be found in smdsForMirrorTskLogBTL.xlsx

**Requirements**

The priority of these requirements is imperative.

*\*\*Constraints*

1. MirrorBTL is coded in C++.  
   Validation: Inspection and Test (Build)
2. The source file for ServiceStationBTL is in compliance with *MTM C++ Source File Standard v4.0*.  
   Validation: Inspection
3. The SMDS conforms to version 4.2 of the standard.

*\*\*Preconditions*

1. The input file is well-formed, follows the input/output requirements, and exists.

*\*\*Invocation*

1. Launch application execution file.  
   Validation: Test

*\*\*Input/Ouput*

1. Input consists of an unknown number of patterns.  
   Validation: Test
2. Patterns start with a positive integer (patternSize) on its own line that signifies the number of rows and columns in the pattern.  
   Validation: Inspection
3. patternSize is between 1 and 10 inclusive.  
   Validation:Test
4. The following patternSize lines are made up of the patterns.  
   Validation: Test
5. Each pattern line has patternSize characters followed by a space and then another patternSize characters.  
   Validation: Test
6. The first set of patternSize characters is the row of the original image.  
   Validation: Test
7. The second set of patternSize characters is the row of the transformed image.  
   Validation: Test
8. The two allowed characters for the images are a period ‘.’ and ‘X’  
   Validation: Test
9. The variable patternNum corresponds to the number of the pattern in the input sequence.  
   Validation: Test
10. The output is a sentence on its own line output to stdout.  
    Validation: Test
    1. If the image was rotated by 90, 180, or 270 degrees “Pattern XX was rotated ZZ degrees.” is output, where XX is the patternNum of the pattern and ZZ is the rotation angle.
    2. If the image was vertically reflected “Pattern XX was reflected vertically.” is output where XX is the patternNum.
    3. If the image was reflected and then rotated, “Pattern XX was reflected vertically and rotated ZZ degrees.” is output, where XX is the patternNum and ZZ is the number of degrees it was reflected.
    4. If no changes were made to the pattern “Pattern XX was preserved.” is output, where XX is the patternNum.
    5. If none of the above is true, “Pattern XX was improperly transformed.” is output, where XX is the patternNum.

*\*\*Postconditons*

1. The input file is still in the system.  
    Validation: Inspection

*\*\*Testing*

1. MirrorBTL will be tested using test scripts and test files.  
   Validation: Inspection
2. MirrorBTL will be randomly tested against another student’s version.Validation: Inspection

*\*\*Inspection*

None

*\*\*Algorithm Correctness Argument*

1. A correctness algorithm will be provided for both the algorithm and the code.Validation: Inspection

*\*\*Sample Runs*

*Sample Run 1:*

5

X...X ....X

.X... ...X.

...X. .X...

..X.X ..X..

....X XX..X

6

....XX X....X

...X.. X.X...

XX..X. .X..X.

..X... ...X.X

...X.. ..X...

..X..X ..X...

2

X. X.

.X .X

4

..X. ...X

XX.. ....

.... XX..

...X ..X.

5

X.... .X...

.X... ..X..

.X... ..X..

...X. ....X

....X X....

4

.X.. ..X.

.X.X X...

.... ..XX

..X. ....

2

.. XX

XX ..

Then stdout should be

Pattern 1 was rotated 90 degrees.

Pattern 2 was rotated 270 degrees.

Pattern 3 was preserved.

Pattern 4 was reflected vertically.

Pattern 5 was improperly transformed.

Pattern 6 was reflected vertically and rotated 270 degrees.

Pattern 7 was rotated 180 degrees.

**Input Space Partitioning**

1. There is one pattern in the input.
   1. patternSize is 1.
      1. The pattern is preserved.
      2. The pattern was improper.
   2. patternSize is 2.
      1. The pattern is preserved.
      2. The pattern is a rotation.
      3. The pattern is a reflection.
      4. The pattern is a reflection and a rotation.
      5. The pattern was improper.
   3. patternSize is greater than 2 but at most 10.
      1. The pattern is preserved.
      2. The pattern is a rotation.
      3. The pattern is a reflection.
      4. The pattern is a reflection and a rotation.
      5. The pattern was improper.
2. There are two patterns in the input.

Two patterns that are variations on (1).

1. There are more than two patterns in the input.

Several patterns that are variations on (1).

**Test Cases**

|  |  |  |
| --- | --- | --- |
| **Input Space Partition Summary** | **Input file for each test case** | **Output file for this test case** |
| 1.1.1  One pattern  patternSize is 1  Pattern is preserved | *mirrorT01in:*  1  . . | *mirrorT01out:*  Pattern 1 was preserved. |
| 1.1.2  One pattern  patternSize is 1  Pattern is improper | *mirrorT02in:*  1  . X | *mirrorT02out:*  Pattern 1 was improperly transformed. |
| 1.2.1  One pattern  patternSize is 2  Pattern is preserved | *mirrorT03in:*  2  .X .X  X. X. | *mirrorT03out:*  Pattern 1 was preserved. |
| 1.2.2  One pattern  patternSize is 2  Pattern is a rotation | *mirrorT04in:*  2  .X X.  X. .X | *mirrorT04out:*  Pattern 1 was rotated 90 degrees. |
| 1.2.3  One pattern  patternSize is 2  Pattern is a reflection | *mirrorT05in:*  2  .. XX  XX .. | *mirrorT05out:*  Pattern 1 was reflected vertically. |
| 1.2.4  One pattern  patternSize is 2  Pattern is a reflection and rotation | *mirrorT06in:*  2  .. X.  XX X. | *mirrorT06out:*  Pattern 1 was reflected vertically and rotated 90 degrees. |
| 1.2.5  One pattern  patternSize is 2  Pattern is improper | *mirrorT07in:*  2  .. XX  .. XX | *mirrorT07out:*  Pattern 1 was improperly transformed. |
| 1.3.1  One pattern  patternSize is greater than 2 and no more than 10  Pattern is preserved | *mirrorT08in:*  5  ..... .....  XXXXX XXXXX  .X.X. .X.X.  X.X.X X.X.X  XX.XX XX.XX | *mirrorT08out:*  Pattern 1 was preserved*.* |
| 1.3.2  One pattern  patternSize is greater than 2 and no more than 10  Pattern is a rotation | *mirrorT09in:*  8  X..XX..X XXXXXXXX  XXXXXXXX ..XX..XX  XXXXXXXX XX..XX..  XXXXXXXX XXXXXXXX  XXXXXXXX XXXXXXXX  ..XX..XX XXXXXXXX  XX..XX.. XXXXXXXX  XXXXXXXX X..XX..X | *mirrorT09out:*  Pattern 1 was rotated 180 degrees. |
| 1.3.3  One pattern  patternSize is greater than 2 and no more than 10  Pattern is a reflection | *mirrorT10in:*  4  X.X. .X.X  XXXX ....  .... XXXX  .X.X X.X. | *mirrorT10out:*  Pattern 1 was reflected vertically. |
| 1.3.4  One pattern  patternSize is greater than 2 and no more than 10  Pattern is a reflection and a rotation | *mirrorT11in:*  4  X.X. .X.X  XXXX XXXX  .... ....  .X.X X.X. | *mirrorT11out:*  Pattern 1 was reflected vertically and rotated 180 degrees. |
| 1.3.5  One pattern  patternSize is greater than 2 and no more than 10  Pattern is improper | *mirrorT12in:*  6  X.X.X. X.X.X.  .X.X.X .X.X.X  X.X.X. X.X.X.  XXXXXX XXXXXX  ...... ......  XXXXXX XXXXX. | *mirrorT12out:*  Pattern 1 was improperly transformed. |
| 2  Two patterns | *mirrorT13in:*  1  . .  1  . X | *mirrorT13out:*  Pattern 1 was preserved.  Pattern 2 was improperly transformed. |
| 2  Two Patterns | *mirrorT14in:*  2  .X .X  X. X.  2  .X X.  X. .X | *mirrorT14out:*  Pattern 1 was preserved.  Pattern 2 was rotated 90 degrees. |
| 2  Two Patterns | *mirrorT15in:*  2  .. XX  XX ..  2  .. X.  XX X. | *mirrorT15out:*  Pattern 1 was reflected vertically.  Pattern 2 was reflected vertically and rotated 90 degrees. |
| 2  Two Patterns | *mirrorT16in:*  2  .. XX  .. XX  5  ..... .....  XXXXX XXXXX  .X.X. .X.X.  X.X.X X.X.X  XX.XX XX.XX | *mirrorT16out:*  Pattern 1 was improperly transformed.  Pattern 2 was preserved*.* |
| 2  Two Patterns | *mirrorT17in:*  8  X..XX..X XXXXXXXX  XXXXXXXX ..XX..XX  XXXXXXXX XX..XX..  XXXXXXXX XXXXXXXX  XXXXXXXX XXXXXXXX  ..XX..XX XXXXXXXX  XX..XX.. XXXXXXXX  XXXXXXXX X..XX..X  4  X.X. .X.X  XXXX ....  .... XXXX  .X.X X.X. | *mirrorT17out:*  Pattern 1 was rotated 180 degrees.  Pattern 2 was reflected vertically. |
| 3  More than two patterns | *mirrorT18in:*  8  X..XX..X XXXXXXXX  XXXXXXXX ..XX..XX  XXXXXXXX XX..XX..  XXXXXXXX XXXXXXXX  XXXXXXXX XXXXXXXX  ..XX..XX XXXXXXXX  XX..XX.. XXXXXXXX  XXXXXXXX X..XX..X  5  ..... .....  XXXXX XXXXX  .X.X. .X.X.  X.X.X X.X.X  XX.XX XX.XX  2  .. XX  XX .. | *mirrorT18out:*  Pattern 1 was rotated 180 degrees.  Pattern 2 was preserved*.*  Pattern 3 was reflected vertically. |
| 3  More than two patterns | *mirrorT19in:*  4  X.X. .X.X  XXXX ....  .... XXXX  .X.X X.X.  4  X.X. .X.X  XXXX XXXX  .... ....  .X.X X.X.  6  X.X.X. X.X.X.  .X.X.X .X.X.X  X.X.X. X.X.X.  XXXXXX XXXXXX  ...... ......  XXXXXX XXXXX. | *mirrorT19out:*  Pattern 1 was reflected vertically.  Pattern 2 was reflected vertically and rotated 180 degrees.  Pattern 3 was improperly transformed. |
| 3  More than two patterns | *mirrorT20in:*  1  . .  1  . X  2  .X .X  X. X.  2  .X X.  X. .X  2  .. XX  XX ..  2  .. X.  XX X.  2  .. XX  .. XX | *mirrorT20out:*  Pattern 1 was preserved.  Pattern 2 was improperly transformed.  Pattern 3 was preserved.  Pattern 4 was rotated 90 degrees.  Pattern 5 was reflected vertically.  Pattern 6 was reflected vertically and rotated 90 degrees.  Pattern 7 was improperly transformed. |
| 3  More than two patterns | *mirrorT21in:*  1  . .  1  . X  2  .X .X  X. X.  2  .X X.  X. .X  2  .. XX  XX ..  2  .. X.  XX X.  2  .. XX  .. XX  5  ..... .....  XXXXX XXXXX  .X.X. .X.X.  X.X.X X.X.X  XX.XX XX.XX  8  X..XX..X XXXXXXXX  XXXXXXXX ..XX..XX  XXXXXXXX XX..XX..  XXXXXXXX XXXXXXXX  XXXXXXXX XXXXXXXX  ..XX..XX XXXXXXXX  XX..XX.. XXXXXXXX  XXXXXXXX X..XX..X  4  X.X. .X.X  XXXX ....  .... XXXX  .X.X X.X.  4  X.X. .X.X  XXXX XXXX  .... ....  .X.X X.X.  6  X.X.X. X.X.X.  .X.X.X .X.X.X  X.X.X. X.X.X.  XXXXXX XXXXXX  ...... ......  XXXXXX XXXXX. | *mirrorT21out:*  Pattern 1 was preserved.  Pattern 2 was improperly transformed.  Pattern 3 was preserved.  Pattern 4 was rotated 90 degrees.  Pattern 5 was reflected vertically.  Pattern 6 was reflected vertically and rotated 90 degrees.  Pattern 7 was improperly transformed.  Pattern 8 was preserved*.*  Pattern 9 was rotated 180 degrees.  Pattern 10 was reflected vertically.  Pattern 11 was reflected vertically and rotated 180 degrees.  Pattern 12 was improperly transformed. |

**Design**

*Significant Data Items/Structures*

int[][] transformationMatrix

Rows relate to the various transformations

Columns relates to the following

1st column: invertX – If 1, read the X values from high to low

2nd column: invertY - If 1, read the Y values from high to low

3rd column: invertXYPriority – If 1, iterate y then x instead of x and then y

Preserve [[0,0,0],

90 degree [1,0,1],

180 Degree [1,1,0],

270 degree [0,1,1],

Reflection [0,1,0],

Reflection and 90 Degree [0,0,1],

Reflection and 180 Degree [1,0,0],

Reflection and 270 Degree [1,1,1]]

The order of rows also accounts for the amount of work.

int[][] originalImage;

int[][] newImage;

char originalValue, transformedValue;

int NUM\_TRANSFORMATIONS = 8;

int MAX\_SIZE = 10;

int patternNum, patternSize, transformedXNdx, transformedYNdx;

Boolean isValidTransformation;

String[] transformationMessage;

*Solution Analys*is

The problem will be solved by looking at the input array and by looking at the indices in different orders, checking to see if there is a valid transformation that has been applied to create the output array.

Given Input

2

.X X.

X. .X

This is a 90 degree rotation.

|  |  |
| --- | --- |
| Original Image | New Image |
| .X  X. | X.  .X |

Origin

X.

.X

This works because if we know where each transformations origin is, and how to iterate over the x and y indices then we can read the new image and check each index against the original image. The benefit of this method is that we can stop once we decide that the images do not follow the transformation we are checking. transformationMatrix holds the information about how to read the newImage array to compare it against the originalImage.

*Algorithms*

A00 **Define**:

int[][] transformationMatrix

int[][] originalImage;

int[][] newImage;

char originalValue, transformedValue;

int NUM\_TRANSFORMATIONS = 8;

int patternNum, patternSize, transformedXNdx,

transformedYNdx;

Boolean isValidTransformation;

String[] transformationMessage;

A01 **While**(stdin is not empty){

A02 **Read** patternSize from stdin;

A03 **For**(lineNdx iterates from 0 to patternSize){

A04 **Read** originalImage and newImage from stdin;

}//**For**

A05 **For**(transformationNdx iterates from 0 to NUM\_TRANSFORMATIONS -1){

A06 **Set** isValidTransformation to true;

A07 **For**(xNdx iterates from 0 to patternSize -1){

A08 **For**(yNdx iterates from 0 to patternSize -1){

A09 **Set** originalValue to newImage[xNdx][yNdx];

**Set** transformedXNdx to xNdx;

**Set** transformedYNdx to yNdx;

A10 **If**(transformationMatrix[transformationNdx][2] is

equal to 1){

A11 **Set** transformedXNdxtoyNdx;

**Set** transformedYNdxtoxNdx;

**}**

A12 **If**(transformationMatrix[transformationNdx][0] is

equal to 1){

A13 **Set** transformedXNdx

to patternSize -1 - transformedXNdx;

}//**If**

A14 **If**(transformationMatrix[transformationNdx][1] is

equal to 1){

A15 **Set** transformedYNdx

to patternSize -1 -transformedYNdx;

}//**If**

A16 **Set** transformedValue to

originalImage[transformedXNdx][transformedYNdx];

A17 **If**(transformedValue is not equal to originalValue){

A18 **Set** isValidTransformation to false;

**Break**;

}//**If**

}//**For** yNdx

A19 **If**(isValidTransformation is false){

A20 **Break**;

}//**If**

}//**For** xNdx

A21 **If**(isValidTransformation is true){

A22 **Display** “Pattern ” + patternNum + “ ” +

transformationMessage[transformationNdx];

A23 **Break**;

}//**If**

}//**For** transformationNdx

A24 **Set** patternNum to patternNum + 1;

}//**While**

A25**Return***;*

Algorithm Correctness Argument

1. A01 continues reading until the end of the file
2. patternSize is read in by A02
3. originalImage and newImage both are sized when instantiated in A00 to hold   
    up to a pattern of size ten
4. A03 reads in patternSize lines of input for the patterns
5. A04 reads in one line of the patterns
6. A04 reads the originalImage first
7. After reading the first patternSize characters and consuming the space A04   
    reads the pattern information into newImage
8. The valid characters are added to the arrays.
9. pattternNum is started in A00 with the value 1, patternNum is incremented   
    in A25 by one to increment the number to print, and then patternNum is   
    displayed on A22 so that the output contains the patternNum
10. A22 prints out the statement

**Code**

*Please see MirrorBTL.cpp for the up-to-date code file.*

**Requirements Trace**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Function(s)** | **Code Element** |
| R06 | main | L01 |
| R07 | main | L02 |
| R08 | main | L00 |
| R09 | main | L03 |
| R10 | main | L04 |
| R11 | main | L04 |
| R12 | main | L04 |
| R13 | main | L04 |
| R14 | main | L00,L22,L25 |
| R15 | main | L22 |

**Code Correctness Arguments**

Code Correctness Argument for function main:

CA01 While loop iterates until the stdin is empty

CA02 The first line is read in and contains the patternNum

CA03 lineNdx iterates from 0 to patternSize -1

CA04 Both images are read in from stdin

CA05 The loop iterates through possible transformations

CA06 isValidTransformation is set to true

CA07 xNdx is iterated from 0 to patternSize - 1

CA08 yNdx is iterated from 0 to patternSize - 1

CA09 originalValue, newImage, transformedXNdx, and transformedYNdx are set

CA10 Value from transformationMatrix is checked

CA11 transformedXNdx and transformedYNdx are set

CA12 transformationMatrix value is checked

CA13 transformedXNdx is set

CA14 transformationMatrix is checked

CA15 transformedYNdx is set

CA16 transformedValue is set

CA17 transformedValue is checked against originalValue

CA18 isValidTransformation is set

CA19 isValidTransformation is checked

CA20 Loop is broken

CA21 isValidTransformation is checked

CA22 Output is sent to stdout

CA23 Loop is broken

CA24 patternNum is incremented

CA25 Program returns at end.

**Test Directories, Files, and Scripts/Scenarios**

For documentation on testing script please read the comments at the top of mirrorUnitTestBTL.sh.

**Random Test Generation**

For documentation on the random test script please read the comments at the top of mirrorRandomTestBTL.sh.

**Performance Test Procedure**

No performance tests were performed.

**Inspection Report(s)**

No inspection was performed.

**Deliverables**

1. MirrorSMDS\_BTL.docx
2. mirrorLogBTL.xlsx
3. mirrorUnitTestsBTL.zip
   * mirrorBTL.cpp
   * mirrorUnitTestBTL.sh
   * unit\_tests directory with the test cases
     + mirrorInXX.txt where XX is a two digit number between 1 and 21 inclusive.
     + mirrorOutXX.txt where XX is a two digit number between 1 and 21 inclusive.
   * mirrorUnitTestRsltsBTL.txt
4. mirrorCoverageTestsBTL.zip
   * mirrorBTL.cpp
   * mirrorCoverageTestBTL.sh
   * mirrorCoverageTestRsltsBTL.txt
5. mirrorRandomTestsBTL.zip
   * mirrorBTL.cpp
   * mirrorFMD.cpp
   * testGenerator/mirrorRandomTestBTL.cpp
   * myRandomTestResultsBTL.txt
   * otherRandomTestResultsBTL.txt
   * randomFiles directory with the test cases
     + mirrorInXX.txt where XX is the number of the test case
6. smdsPartOfFinal.docx

**Version History**

Most real world programs will undergo some modification during their lifetime. The initial program version and any subsequent modifications should be documented here.

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Author* | *Comment* |
| 1.0 | 12/13/16 | Trevor Brooks | Passes UVa judge. |
|  |  |  |  |