This document is in compliance with SMDS standard version 4.2

|  |  |
| --- | --- |
| **Description:** | A program that evaluates pairs of square patterns and identifies the minimum transformation applied to go from the original pattern to the transformed pattern. |

**Development Estimates/Actuals**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Item** | **Est Sz** | **Act Sz** |  | **Strt Date** | **Est Cmplt** | **Act Cmplt** |  | **Est Effrt** | **Act Effrt** |
| Reqs | 20 | 18 |  | 12/8/16 | 12/8/16 | 12/8/16 |  | 1.5 | .75 |
| Inpt Partition | 20 | 22 |  | 12/8/16 | 12/8/16 | 12/8/16 |  | 1 | .75 |
| Test Cases | 20 | 17 |  | 12/8/16 | 12/8/16 | 12/8/16 |  | 1 | 1.5 |
| Design |  |  |  |  |  |  |  | 1.5 | .65 |
| # Classes | *1* | 1 |  |  |  |  |  |  |  |
| # Methods | *2* | *5* |  |  |  |  |  |  |  |
| # Dsng Elms | *5* | *3* |  |  |  |  |  |  |  |
| Algor Correct |  |  |  | 12/9/16 | 12/9/16 | 12/9/16 |  | 1.5 | 1.0 |
| Implementatn |  |  |  | 12/9/16 | 12/9/16 | 12/9/16 |  | 2 | 1.0 |
| NCLC |  |  |  |  |  |  |  |  |  |
| Requir. Trace |  |  |  |  |  |  |  |  |  |
| Code Correct |  |  |  | 12/9/16 | 12/9/16 | 12/10/16 |  | 1 | 1.15 |
| Final Test |  |  |  | 12/10/16 | 12/10/16 | 12/12/16 |  | 1 | 1.35 |
| Inspec |  |  |  | 12/12/16 | 12/12/16 | 12/12/16 |  | .5 | 1.5 |
| Wrap up |  |  |  | 12/12/16 | 12/12/16 | 12/12/16 |  | .5 | .25 |
|  |  |  |  |  |  | Total Hrs |  | *11.5 hrs* | *9.05 hrs* |

*In this section you practice an important software engineering skill: estimating sizes, effort, and completion dates. Estimates should be made in this order. These need not be especially exact, but as you work through this sheet at different times you will need to keep an SMDS Task Log (see associated smdsTskLogLFM.xlsx file). You will probably find that you will not work on these sections strictly sequentially, that is, you may find that you need to redo statements in previous sections until you have completed your final code correctness arguments.*

Estimates/Actuals Comments

* *Deliverables must be completed by 12/13/2016.*
* A relatively short development time means my estimates may be overly hasty, will strive to see if they can be met regardless.
* Unit tests ran successfully on the first pass.
* Coverage testing showed 100% of code lines were executed.
* Random Testing ran successfully on the first pass.
* Accepted by UVa on first submission.

**Requirements**

All requirements are of priority level: Imperative

*\*\*Constraints*

1. MirrorRAR will be developed using the *MTM SMDS template version 4.2*Validation: Observation & Inspection
2. MirrorRAR will be coded in C++ and adheres to the *MTM Standard for C++ Source Files*Validation: Observation & Inspection

*\*\*Preconditions*

1. The directory that contains the MirrorRAR executable must also contain an input test file, or multiple files, which details the pattern datasets that must be evaluated.  
   Validation: Observation & Test

*\*\*Invocation*

1. MirrorRAR will be invoked from the Windows command line: *./MirrorRAR*Validation: Observation & Test
2. MirrorRAR will be supplied an input text file input.txt from which to read in pattern datasets.  
   Validation: Guaranteed

*\*\*Input/Ouput*

1. Each input file will contain an **unknown** number of pattern datasets to evaluate.  
   Validation: Guaranteed
2. Each pattern dataset will consist of a single integer, ranging from 1-10, on a single line by itself, denoting the dimensions of the square containing the pattern.  
   Validation: Guaranteed
3. The lines of input following the single integer will contain each line of the original and new (transformed) patterns, in a side-by-side format, separated by one space. Both squares are the size of the single integer read in from R07.  
   Validation: Guaranteed
4. Light squares in each pattern will be represented with a dot “.” and dark squares will be represented with the letter “X”.  
   Validation: Guaranteed
5. The output will be contained in a text file, which will feature a sentence for each pattern evaluation describing the transformed pattern’s relationship with the original.  
   Validation: Observation
6. Each sentence will begin with a pattern ID number (starting with 1) and ending in the relationship representing the minimal amount of work needed to derive the new pattern from the original.  
   Validation: Observation
7. For the purpose of evaluation, only the following transformations shall be considered:
   1. All rotations will be clockwise.
   2. For evaluation, rotations are less “work” than reflections.
   3. 90 Degree Rotation
   4. 180 Degree Rotation
   5. 270 Degree Rotation
   6. Vertical Reflection – The pattern was reflected through a horizontal mirror positioned above the pattern.
   7. Combination – The pattern was subjected to a vertical reflection, followed by one of the possible rotations
   8. Preservation – The original pattern was preserved (no transformation)
   9. Improper – The transformed pattern was not obtained from the original pattern.

*\*\*Postconditons*

1. None.

*\*\*Testing*

1. MirrorRAR will be tested using repeatable test scripts and files. Statement and branch coverage will be determined. Any statements or branches not executed will be explained.  
   Validation: Inspection
2. MirrorRAR will be tested using a random test generator. Any discrepancies will be explained.  
   Validation: Inspection

*\*\*Inspection*

1. An inspection on all completed sections will be performed on a later date.

A final inspection will be done before delivering program.  
Validation: Observation

*\*\*Algorithm Correctness Argument*

1. An algorithm correctness argument is included in this SMDS.  
   Validation: Observation

*\*\*Code Correctness Argument*

1. A code correctness argument is included in this SMDS.  
   Validation: Observation

Sample Runs *Sample Input*

5  
X…X ….X  
.X… …X.  
…X. .X…  
..X.X ..X..  
….X XX..X

2  
X. X.  
.X .X

4  
..X. …X  
XX.. ….  
…. XX..  
…X ..X.

4  
.X.. ..X.  
.X.X X…  
…. ..XX  
..X. ….

2  
.. XX  
XX ..  
  
*Sample Output*

Pattern 1 was rotated 90 degrees.  
Pattern 2 was preserved.  
Pattern 3 was reflected vertically.  
Pattern 4 was reflected vertically and rotated 270 degrees.  
Pattern 5 was rotated 180 degrees.

**Input Space Partitioning**

1. There is one pattern dataset
   1. The dimSize is 1
      1. The pattern was Preserved
      2. The pattern is Improper
   2. The dimSize is greater than 1 and less than 10
      1. The pattern was Preserved
      2. The pattern was Rotated 90°
      3. The pattern was Rotated 180°
      4. The pattern was Rotated 270°
      5. The pattern was Reflected Vertically
      6. The pattern was transformed using a combination of Reflection & Rotation
      7. The pattern was improperly transformed
   3. The dimSize is 10
      1. The pattern was Preserved
      2. The pattern was Rotated 90°
      3. The pattern was Rotated 180°
      4. The pattern was Rotated 270°
      5. The pattern was Reflected Vertically
      6. The pattern was transformed using a combination of Reflection & Rotation
      7. The pattern was improperly transformed
2. There is more than one pattern dataset
   1. Each pattern will be a variation of results found in (1).

**Test Cases**

|  |  |  |
| --- | --- | --- |
| **Input Space Partition Summary** | **Input file for each test case** | **Output file for this test case** |
| 1. 1.1.1   dimSize = 1 | **smtst\_1\_in**  1 X X | **smtst\_1\_out**  Pattern 1 was **preserved**. |
| 2. 1.1.2  dimSize = 1 | **smtst\_2\_in**  1 . X | **smtst\_2\_out**  Pattern 1 was **improperly transformed.** |
| 3. 1.2.1  1 < dimSize < 10 | **smtst\_3\_in**  2 XX XX | **smtst\_3\_out**  Pattern 1 was **preserved.** |
| 4. 1.2.2  1 < dimSize < 10 | **smtst\_4\_in**  5 X…X ….X .X… …X.  …X. .X…  ..X.X ..X..  ….X XX..X | **smtst\_4\_out**  Pattern 1 was **rotated 90 degrees.** |
| 5. 1.2.3  1 < dimSize < 10 | **smtst\_5\_in**  2  .. XX XX .. | **smtst\_5\_out**  Pattern 1 was **rotated 180 degrees.** |
| 6. 1.2.4  1 < dimSize < 10 | **smtst\_6\_in**  6 ….XX X….X …X.. X.X…  XX..X. .X..X.  ..X… …X.X  …X.. ..X…  ..X..X ..X… | **smtst\_6\_out**  Pattern 1 was **rotated 270 degrees.** |
| 7. 1.2.5  1 < dimSize < 10 | **smtst\_7\_in**  4 ..X. …X XX.. ….  …. XX..  …X ..X. | **smtst\_7\_out**  Pattern 1 was **reflected vertically.** |
| 8. 1.2.6  1 < dimSize < 10 | **smtst\_8\_in**  4  .X.. ..X.  .X.X X…  …. ..XX  ..X. …. | **smtst\_8\_out**  Pattern 1 was **reflected vertically and rotated 270 degrees.** |
| 9. 1.2.7  1 < dimSize < 10 | **smtst\_9\_in**  5  X…. .X…  .X… ..X..  .X… ..X..  …X. ….X  ….X X…. | **smtst\_9\_out**  Pattern 1 was **improperly transformed.** |
| 10. 1.3.1  dimSize = 10 | **smtst\_10\_in**  10  X……… X……… ..X..X…. ..X..X….  X….X…X X….X…X  ….X….. ….X…..  ………. ……….  XXX……. XXX…….  X.X……. X.X…….  …….XXX …….XXX  X.X.X.X… X.X.X.X…  …X…X.. …X…X.. | **smtst\_10\_out**  Pattern 1 was **preserved.** |
| 11. 1.3.2  dimSize = 10 | **smtst\_11\_in**  10  X……… .X.XX..X.X ..X..X…. ….X…..  X….X…X .X.XX…X.  ….X….. X………  ………. .X….X…  XXX……. …….XX.  X.X……. .X……..  …….XXX X.X…….  X.X.X.X… ..X…….  …X…X.. ..X….X.. | **smtst\_11\_out**  Pattern 1 was **rotated 90 degrees.** |
| 12. 1.3.3  dimSize = 10 | **smtst\_12\_in**  10  X……… ..X…X…  ..X..X…. …X.X.X.X  X….X…X XXX…….  ….X….. …….X.X  ………. …….XXX  XXX……. ……….  X.X……. …..X….  …….XXX X…X….X  X.X.X.X… ….X..X..  …X…X.. ………X | **smtst\_12\_out**  Pattern 1 was **rotated 180 degrees.** |
| 13. 1.3.4  dimSize = 10 | **smtst\_13\_in**  10  X……… ..X….X.. ..X..X…. …….X..  X….X…X …….X.X  ….X….. ……..X.  ………. .XX…….  XXX……. …X….X.  X.X……. ………X  …….XXX .X…XX.X.  X.X.X.X… …..X….  …X…X.. X.X..XX.X. | **smtst\_13\_out**  Pattern 1 was **rotated 270 degrees.** |
| 14. 1.3.5  dimSize = 10 | **smtst\_14\_in**  10  XX…….. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. XX…….. | **smtst\_14\_out**  Pattern 1 was **reflected vertically.** |
| 15. 1.3.6  dimSize = 10 | **smtst\_15\_in**  10  XX…….. X..…….  ………. X..…….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ..…….. | **smtst\_15\_out**  Pattern 1 was **reflected vertically and rotated 90 degrees.** |
| 16. 1.3.7  dimSize = 10 | **smtst\_16\_in**  10 XX…….. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ….XX….  ………. ……….  ………. ……….  ………. ……….  ………. ……….  ………. ..…….. | **smtst\_16\_out**  Pattern 1 was **improperly transformed.** |
| 17. 2.2.1  dimSize = 5 & 3 | **smtst\_17\_in**  5 X…X ….X .X… …X.  …X. .X…  ..X.X ..X..  ….X XX..X  3  X.X ..X  … X..  X.. ..X | **smtst\_17\_out**  Pattern 1 was **rotated 90 degrees.**  Pattern 2 was **improperly transformed.** |

**Design**

*Significant Data Items/Structures*

* char pattern[3][12][12]
* int dimSize
* int dimSize2
* int patternNmbr
* bool same()
* bool rotate()
* void reflect()

*Solution Analys*is

The main() function contains a while loop that runs while cin can read in from dimSize number of lines of text. Using an inner for loop, for each line of text, it reads in and set the values of pattern[0][] (the original) and pattern[1][] (the transformed). It then calls the evaluate() function as it couts for each pattern evaluation, incrementing patternNmbr to the appropriate number.

The *same()* function uses two for loops (both looping from 0 to dimSize) to compare if pattern[0][ndx][ndy] and pattern[1][ndx][ndy] are the same at each pair of indexes, returning a true value if so.  
  
The *rotate()* function uses two for loops (both looping from 0 to dimSize) to compare if pattern[0][ndx][ndy] and pattern[2][ndx][ndy] are the same at each pair of indexes, returning a true value if so.

The reflect()function uses two for loops (one looping from ndx *=* 0 to dimSize, and the other looping from ndy = 0 to dimSize2) to swap the values of pattern[2][ndx][ndy] and pattern[2][dimSize – ndx - 1][ndy].

The evaluate()function first calls same() to check if the two patterns are the same. If not it then uses two for loops (both looping from 0 to dimSize) to perform a 90-degree rotation by setting the values of pattern[1][ndy][dimSize – ndx – 1] (the transformed pattern) to a new placeholder matrix pattern[2][ndx][ndy]. It then calls the rotate() function to see if it evaluates as true.  
  
If rotate() returns false, evaluate() then performs a 180-degree rotation by setting the values of pattern[1][ dimSize – ndx – 1][dimSize – ndy – 1] (the transformed pattern) to a new placeholder matrix pattern[2][ndx][ndy]*.* It then calls rotate(), to see if it evaluates as true.  
  
If rotate() returns false, evaluate() then performs a 270-degree rotation by setting the values of pattern[1][ dimSize – ndy – 1][ndx] (the transformed pattern) to a new placeholder matrix pattern[2][ndx][ndy]*.* It then calls the rotate() function to see if it evaluates as true.

If rotate() returns false, evaluate() then determines a new dimSize2variable by dividing dimSize in half. This dimSize2 variable gets used in one reflect()’s for loops, a function which has yet to be called on during runtime. Now, evaluate()performs the same 3 previous rotation tests however, it now also calls on reflect(), and then rotate(),to test the combination of vertical reflection and rotations.

If none of these conditions are satisfied, the algorithm returns an improper transformation.

*Algorithm*

A00 **Define & Set** char pattern[3][12][12]

A01 **Define & Set** int dimSize, dimSize2, patternNmbr = 0

A02 main(){

A03 **While**( cin >> dimSize ){

A04 **For**( ndx from 0 to dimSize ){

A05 cin >> pattern[0][ndx] >> pattern[1][ndx]

}//For

A06 cout << “Pattern” << ++patternNmbr << “was” <<

evaluate() << endl

}//While

}//main

A07 evaluate(){

A08 **If**( same() ) **Return** “preserved.”

//If

A09 **For**(ndx from 0 to dimSize)

A10 **For**(ndy from 0 to dimSize)

A11 **Set** pattern[2][ndx][ndy] =

pattern[1][ndy][dimSize- ndx - 1]

//For

//For

A12 **If**( rotate() ) **Return** "rotated 90 degrees."

//If

A13 **For**(ndx from 0 to dimSize)

A14 **For**(ndy from 0 to dimSize)

A15 **Set** pattern[2][ndx][ndy] =

pattern[1][dimSize-ndx-1][dimSize–ndy-1]

//For

//For

A16 **If**( rotate() ) **Return** "rotated 180 degrees."

//If

A17 **For**(ndx from 0 to dimSize)

A18 **For**(ndy from 0 to dimSize)

A19 **Set** pattern[2][ndx][ndy] =

pattern[1][dimSize-ndy-1][ndx]

//For

//For

A20 **If**( rotate() ) **Return** "rotated 270 degrees."

//If

A21 **For**(ndx from 0 to dimSize)

A22 **For**(ndy from 0 to dimSize)

A23 **Set** pattern[2][ndx][ndy] =

pattern[1][dimSize – ndx - 1][ndy]

//For

//For

A24 **If**( rotate() ) **Return** "reflected vertically."

//If

A25 dimSize2 = dimSize/2

A26 **For**(ndx from 0 to dimSize)

A27 **For**(ndy from 0 to dimSize)

A28 **Set** pattern[2][ndx][ndy] =

pattern[1][ndy][dimSize-ndx-1]

//For

//For

A29 **Call** reflect()

A30 **If**( rotate() ) **Return** "reflected vertically and rotated 90 degrees."

//If

A31 **For**(ndx from 0 to dimSize)

A32 **For**(ndy from 0 to dimSize)

A33 **Set** pattern[2][ndx][ndy] =

pattern[1][dimSize-ndx-1][dimSize-ndy-1]

//For

//For

A34 **Call** reflect()

A35 **If**( rotate() ) **Return** "reflected vertically and rotated 180 degrees."

//If

A36 **For**(ndx from 0 to dimSize)

A37 **For**(ndy from 0 to dimSize)

A38 **Set** pattern[2][ndx][ndy] =

pattern[1][dimSize-ndy-1][ndx]

//For

//For

A39 **Call** reflect();

A40 **If(** rotate() ) **Return** "reflected vertically and rotated 270 degrees."

//If

A41 **Return** "improperly transformed.";

}//evaluate()

A42 same(){

A43 **For**(ndx from 0 to dimSize)

A44 **For**(ndy from 0 to dimSize)

A45 **If**( pattern[0][ndx][ndy] != pattern[1][ndx][ndy] ) A46 **Return** false;

//If

//For

//For

A47 **Return** true;

}//same()

A48 rotate(){

A59 **For**(ndx from 0 to dimSize)

A50 **For**(ndy from 0 to dimSize)

A51 **If**( pattern[0][ndx][ndy] != pattern[2][ndx][ndy]) A52 **Return** false;

//If

//For

//For

A53 **Return** true;

}//rotate()

A54 reflect(){

A55 **For**(ndx from 0 to dimSize2)

A56 **For**(ndy from 0 to dimSize)

A57 **Call** swap(pattern[2][ndx][ndy],

pattern[2][dimSize-ndx-1][ndy])

}//reflect

Algorithm Correctness Argument

* Note: R01 – R06, R09 are not applicable in correctness argument.
* CR07 by A00 – pattern[][][] arrays are initialized to be of size [3][12][12]
  + (2 extra array indexes to handle rearrangement of array values)
* CR08 by A04 and A05, structure of for-loop and read in commands expect the specified input format
* CR10 by A06, cout command is as specified in problem statement.
* CR11 by A06, cout command is as specified in problem statement.
* CR12.a by A11, A15, A19, A28, A33, A38 – all rotations are clockwise
* CR12.b by A16, A24 – when evaluating, 180–degree rotation will be valid before reflection.
* CR12.c by A12
* CR12.d by A16
* CR12.e by A20
* CR12.f by A24
* CR12.g by A30, A35, A40
* CR12.h by A08
* CR12.i by A41

**Code**

Refer to MirrorRAR.cpp located in this directory. Source file complies with MTM Standard for C++ Source Files (ver. 3.2)

Requirements Trace

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Classes and Objects** | **Method and Design Element** |
| R07 | 1. MirrorRAR body | L00 - char pattern[3][12][12] |
| R08 | 1. main() | L04, L05 |
| R10 | 1. main() | L06 - cout |
| R11 | 1. main() | L06 - cout |
| R12.a | 1. evaluate() | L11, L15, L19, L28, L33, L38 |
| R12.b | 1. evaluate() | L16, L24 |
| R12.c | 1. evaluate() | L12 |
| R12.d | 1. evaluate() | L16 |
| R12.e | 1. evaluate() | L20 |
| R12.f | 1. evaluate() | L24 |
| R12.g | 1. evaluate() | L25, L30, L35 |
| R12.h | 1. evaluate() | L08 |
| R12.i | 1. evaluate() | L41 |

Code Correctness Argument

CA00 by L00

CA01 by L01

CA02 by L02

CA03 by L03

CA04 by L04

CA05 by L05

CA06 by L06

CA07 by L07

CA08 by L08

CA09 by L09

CA10 by L10

CA11 by L11

CA11 by L12

CA13 by L13

CA14 by L14

CA15 by L15

CA16 by L16

CA17 by L17

CA18 by L18

CA19 by L19

CA20 by L20

CA21 by L21

CA22 by L22

CA23 by L23

CA24 by L24

CA25 by L25

CA26 by L26

CA27 by L27

CA28 by L28

CA29 by L29

CA30 by L30

CA31 by L30

CA32 by L30

CA33 by L30

CA34 by L30

CA35 by L30

CA36 by L30

CA37 by L30

CA38 by L30

CA39 by L30

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA40 by L40

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA50 by L50

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

CA60 by L60

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

There exist three separate test directories:

* UnitTesting
  + MirrorRAR.cpp file
  + The input files (formatted: smtst\_\*\_in.txt)
  + The output files (formatted: smtst\_\*\_out.txt)
  + The script to run the tests: MirrorRARtestRun.sh
  + The results file: MirrorRARtestRslts.txt
* CoverageTesting
  + MirrorRAR.cpp file
  + the input files (formatted: smtst\_\*\_in.txt)
  + the script to run the tests mirrorCoverageTestRAR.sh
  + the results file: mirrorCoverageTestRsltsRAR.txt
* RandomTesting
  + MirrorRAR.cpp file
  + MirrorAMD.cpp file (program to compare results against)
  + Random test generator:
  + The randomly generated input files (formatted: smtst\_\*\_in.txt)
  + The script to run the test: mirrorRARrandomTestRun.sh
  + The results file: MirrorRARrandomTestRslts.txt

**Random Test Generation**

Random tests were generated by the mirrorRandomTestGenRAR.cpp file, found in the RandomTesting directory.

**Performance Test Procedure**

No performance testing was done, not applicable.

**Inspection Report(s)**

The inspection report will be completed by Verification & Validation (V&V) team at a later date.

**Deliverables**

The main directory contains

* The MirrorRAR.cpp file
* The SMDS file
* The SMDS project Task Log
* The required assignment sheet.

There also exist three separate sub-directories:

* UnitTesting
  + MirrorRAR.cpp file
  + The input files (formatted: smtst\_\*\_in.txt)
  + The output files (formatted: smtst\_\*\_out.txt)
  + The script to run the tests: MirrorRARtestRun.sh
  + The results file: MirrorRARtestRslts.txt
* CoverageTesting
  + MirrorRAR.cpp file
  + the input files (formatted: smtst\_\*\_in.txt)
  + the script to run the tests mirrorCoverageTestRAR.sh
  + the results file: mirrorCoverageTestRsltsRAR.txt
* RandomTesting
  + MirrorRAR.cpp file
  + MirrorAMD.cpp file (program to compare results against)
  + Random test generator:
  + The randomly generated input files (formatted: smtst\_\*\_in.txt)
  + The script to run the test: mirrorRARrandomTestRun.sh
  + The results file: MirrorRARrandomTestRslts.txt

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Author* | *Comment* |
| 1.1 | 12/08/16 | Alex Reid | Requirements completed |
| 1.2 | 12/09/16 | Alex Reid | Implementation completed |
| 1.3 | 12/12/16 | Alex Reid | All tests completed and passed. |