**SIT323 Cloud Application Development**

**Trimester 2, 2018**

**Assessment Task 1 – Programming Task 1**

Arran (Musa) Fletcher & Thomas Halliday

216107465 &

**Contents**

1. Introduction 999
2. Test Scenario 1 999
3. Test Scenario 2 999
4. Test Scenario 3 999
5. Test Scenario 4 999
6. Test Scenario 5 999
7. Test Scenario 6 999
8. Test Scenario 7 999
9. Test Scenario 8 999
10. Test Scenario 9 999
11. Test Scenario 10 999
12. Test Scenario 11 999
13. Test Scenario 12 999
14. **Introduction**

This document will detail the justification and design of unit tests for the SIT323 Assignment 1 Crozzle software program.

The SIT323 Assignment 1 Crozzle software is being updated by the developers involved with this project to accept new reformatted configuration and data files based on the existing data files used by the software.

These tests are designed and performed for quality assurance, to test the performance and functionality of the software being updated and redesigned, to identify and resolve any issues during development of the program before the project is deployed and submitted for grading.

These tests will be used to ensure the program functions as expected, fulfils the requirements expected of the newly updated program, and can handle data files with illegal, missing, and invalid data with detailed reporting while avoiding program crashes.

1. **Test Scenario 1**
   1. Justifications
      1. Test Case 1.1

This test case will test for a valid boolean input, validated as a boolean, to ensure the program checks valid boolean inputs with a correct response.

* + 1. Test Case 1.2

This test case will test for an invalid input, validated as a boolean, to test for the functions response to an invalid input and ensure the program handles the exception properly without crashing.

* 1. Learning

The program is able to successfully differentiate between a boolean and non-boolean value and return a response to user if an error is detected.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 2**
   1. Justifications
      1. Test Case 1.1

This test case will test for a valid integer, validated as an Int32 class input, to ensure the program correctly validates legal integers for configuration and calculation.

* + 1. Test Case 1.2

This test case will test for a legal integer, which is invalid with the Crozzle configuration, to return an IsInt32 response. This is to test the programs response to a legal, but invalid, data entry and to respond with an appropriate error message while still processing any valid data.

* + 1. Test Case 1.2

This test case will test for a non-integer data entry in the IsInt32() function to test the programs response to an invalid and illegal data entry. The goal for this test case is to respond with an appropriate error message and continue processing after the handling of illegal data.

* 1. Learning

The program can detect a valid integer value, and a non-valid integer value. If an invalid integer value is gven the program will return an error response to the user successfully. If a valid integer is given but the Crozzle is invalid the program will respond with a configuration error.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 3**
   1. Justifications
      1. Test Case 1.1

This test case will pass a valid hex code to the program for hex code validation to test validation of a legal data entry.

* + 1. Test Case 1.2

This test case will pass a valid short form hex code, with three characters after the hash as opposed to a full six, to test validation of a different form of valid hex code.

* + 1. Test Case 1.2

This test case will pass an invalid hex code with missing data to the program to test the programs response to invalid hex code data and the exception handling of the program.

* 1. Learning

The program parses and validate full 6 character hex codes and successfully invalidates and responds to invalid hex codes. The program failed to validate short form 3character hex codes and was updated to include this capability.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 4**
   1. Justifications
      1. Test Case 1.1

This test case will pass a valid series of KeyValues to test the programs ability to parse valid key values.

* + 1. Test Case 1.2

This test case pass a valid series of KeyValues with duplicates to test the programs response to valid but duplicate KeyValue values.

* + 1. Test Case 1.2

This test case will pass a series of invalid and illegal entries to be parsed by the KeyValue class and test the programs response to data that cannot be parsed successfully.

* 1. Learning

The program can successfully parse valid KeyValue entries, duplicate KeyValue entries don’t return an error but the last duplicate entry will overwrite any previous values. The program successfully invalidates and returns errors for invalid KeyValue data.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 5**
   1. Justifications
      1. Test Case 1.1

This test case will pass a valid Crozzle with a valid configuration to be scored to check the programs functionality when scoring a valid Crozzle.

* + 1. Test Case 1.2

This test case will pass legal data with a Crozzle file that has invalid data according to the configuration file rules. This will test the programs scoring response to valid data but an invalid Crozzle.

* + 1. Test Case 1.2

This test case will pass invalid and illegal data to test the programs response to scoring an invalid and incomplete Crozzle.

* 1. Learning

The program successfully scores a valid Crozzle and displays invalid for both scenarios; invalid and illegal data, and valid data but an invalid Crozzle.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 6**
   1. Justifications
      1. Test Case 1.1

This test case will pass a list of valid words with no duplicates to check the programs response to a valid word list without duplicates or errors.

* + 1. Test Case 1.2

This test case will test the programs response to a valid word list that has several duplicates with an expectation to return an error for each duplicate to the user.

* + 1. Test Case 1.2

This test case will test the programs response when checking for duplicates in a word list containing invalid and illegal data. The program should return an invalid word list error and only find duplicates if they are a valid entry.

* 1. Learning

The program successfully responds to a valid word list without duplicates, the program reports errors for each duplicate found in a valid word list, and successfully invalidates an invalid word list.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 7**
   1. Justifications
      1. Test Case 1.1

This test case will pass valid data to the program to test the programs response to validating a valid and legal Crozzle.

* + 1. Test Case 1.2

This test case will pass valid data with an invalid Crozzle configuration to test the programs response to checking an invalid Crozzle and alerting the user accordingly.

* + 1. Test Case 1.2

This test case will pass invalid and illegal data to the program to check the programs response to validation of Crozzle that has invalid, illegal and missing data.

* 1. Learning

The program successfully validates the valid Crozzle and invalidates and reports errors for both the valid data but invalid Crozzle, and the incomplete with invalid data Crozzle.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 8**
   1. Justifications
      1. Test Case 1.1

This test case is designed to test the programs HTML output of a valid Crozzle with specific colour and style rules.

* + 1. Test Case 1.2

This test case is designed to test the program HTML output of an invalid Crozzle with valid data inputs.

* + 1. Test Case 1.2

This test case is designed to test the programs HTML output of an invalid Crozzle with invalid and missing data entries.

* 1. Learning

Regardless of Crozzle validity, if data is present and valid the program will output HTML successfully. If the program is given too much illegal, invalid or missing data the HTML will be unable to output successfully.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 9**
   1. Justifications
      1. Test Case 1.1

This test case is designed to test the programs response to GroupCount for a valid Crozzle.

* + 1. Test Case 1.2

This test case is to test the programs response to GroupCount or an invalid Crozzle.

* + 1. Test Case 1.2

This test case is to test the programs GroupCount functionality for a Crozzle with missing and invalid data and the error response given if encountered.

* 1. Learning

The program is successfully able to return a valid count for any Crozzle with valid data, the GroupCount doesn’t return a valid count if the data files are invalid.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 10**
   1. Justifications
      1. Test Case 1.1

This test case is to check the programs response to validating a valid and legal configuration file.

* + 1. Test Case 1.2

This test case is designed to test the programs response to a valid configuration file containing only logically invalid data when validating the Crozzle.

* + 1. Test Case 1.2

This test case is designed to test the programs response to a configuration file containing invalid, illegal and missing data to check the program correctly reports an invalidation error to users.

* 1. Learning

The program is able to successfully validate legal configuration files with valid data and invalidate configuration files with invalid and illegal data in them returning an error to users.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 11**
   1. Justifications
      1. Test Case 1.1

This test case is designed to check the programs validation response to a valid and legal word list file when parsed.

* + 1. Test Case 1.2

This test case is designed to check the program response to a word list file with no missing or illegal data but with errors within the data given.

* + 1. Test Case 1.2

This test case is to test the programs response to a word list file with invalid and illegal data that can’t be parsed correctly.

* 1. Learning

The program is able to successfully parse and validate the word list files given successfully and return an error for values that are valid but incorrect. The program correctly returns a validation error for the word list file containing invalid data.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

1. **Test Scenario 12**
   1. Justifications
      1. Test Case 1.1

This test is designed to check the programs response to parsing a valid Crozzle file.

* + 1. Test Case 1.2

This test is designed to check the programs response to parsing a Crozzle file with legal but logically invalid data.

* + 1. Test Case 1.2

This test is designed to test the programs parsing of a Crozzle file with invalid, illegal and missing data to return a correct error message to users for a Crozzle file that cannot be parsed.

* 1. Learning

The program is able to parse any Crozzle file with data that can be parsed as legal, and reports an error for parsing a Crozzle file with illegal data present.

* 1. Test Scenario Form

INSERT FORM AND DATA REFERENCES

# Appendix A – Test1.cfg

// Log File Configurations.

LOGFILE

// The default log file name.

DEFAULT="log.txt"

END-LOGFILE

// Letter Sequences Configurations.

// Limits on the number of unique letter sequences in the sequecnes file.

SEQUENCES-IN-FILE

MINIMUM=5

MAXIMUM=500

END-SEQUENCES-IN-FILE

// Crozzle Output Configurations.

CROZZLE-OUTPUT

INVALID-CROZZLE-SCORE="INVALID CROZZLE"

UPPERCASE=true

STYLE="<style> table, td { border: 1px solid black; border-collapse: collapse; } td { width:24px; height:18px; text-align: center; } </style>"

BGCOLOUR-EMPTY-TD=#777777

BGCOLOUR-NON-EMPTY-TD=#ffffff

END-CROZZLE-OUTPUT

// Crozzle Configurations.

// Limits on the size of the crozzle grid.

CROZZLE-SIZE

MINIMUM-ROWS=5

MAXIMUM-ROWS=500

MINIMUM-COLUMNS=5

MAXIMUM-COLUMNS=500

END-CROZZLE-SIZE

// Limits on the number of horizontal letter sequences and

// vertical letter sequences in a crozzle.

SEQUENCES-IN-CROZZLE

MINIMUM-HORIZONTAL=1

MAXIMUM-HORIZONTAL=200

MINIMUM-VERTICAL=1

MAXIMUM-VERTICAL=200

END-SEQUENCES-IN-CROZZLE

// Limits on the number of

// intersecting vertical sequences for each horizontal sequences, and

// intersecting horizontal sequences for each vertical sequences.

INTERSECTIONS-IN-SEQUENCES

MINIMUM-HORIZONTAL=1

MAXIMUM-HORIZONTAL=500

MINIMUM-VERTICAL=1

MAXIMUM-VERTICAL=500

END-INTERSECTIONS-IN-SEQUENCES

// Limits on duplicate letter sequences in the crozzle.

DUPLICATE-SEQUENCES

MINIMUM=0

MAXIMUM=0

END-DUPLICATE-SEQUENCES

// Limits on the number of valid groups.

VALID-GROUPS

MINIMUM=1

MAXIMUM=1

END-VALID-GROUPS

// Scoring Configurations

// Points per letter that is at the intersection of

// a horizontal and vertical sequence within the crozzle.

INTERSECTING-POINTS

A=1

B=2

C=2

D=2

E=1

F=2

G=2

H=2

I=1

J=4

K=4

L=4

M=4

N=4

O=1

P=8

Q=8

R=8

S=8

T=8

U=1

V=16

W=16

X=32

Y=32

Z=64

END-INTERSECTING-POINTS

// Points per letter that is not at the intersection of

// a horizontal and vertical sequence within the crozzle.

NON-INTERSECTING-POINTS

A=0

B=0

C=0

D=0

E=0

F=0

G=0

H=0

I=0

J=0

K=0

L=0

M=0

N=0

O=0

P=0

Q=0

R=0

S=0

T=0

U=0

V=0

W=0

X=0

Y=0

Z=0

END-NON-INTERSECTING-POINTS

# Appendix B – Test1.seq

"[a-zA-Z]{2,}",310,153,11504,11967

AL,10,2,141,153

ALAN,10,4,284,298

ANGELA,10,6,424,440

BETTY,10,5,392,407

BILL,10,4,291,305

BRENDA,10,6,428,444

CHARLES,10,7,514,531

FRED,10,4,289,303

GARY,10,4,307,321

GEORGE,10,6,441,457

GRAHAM,10,6,432,448

HARRY,10,5,390,405

JACK,10,4,281,295

JESSICA,10,7,514,531

JILL,10,4,299,313

JOHNATHON,10,9,681,700

LARRY,10,5,394,409

MARK,10,4,299,313

MARY,10,4,313,327

MATTHEW,10,7,538,555

OSCAR,10,5,376,391

PAM,10,3,222,235

PETER,10,5,384,399

ROBERT,10,6,462,478

ROGER,10,5,383,398

RON,10,3,239,252

RONALD,10,6,448,464

ROSE,10,4,313,327

SUSAN,10,5,394,409

TOM,10,3,240,253

WENDY,10,5,391,406

# Appendix C – Test1.czl

// File dependencies.

FILE-DEPENDENCIES

CONFIG-DATA=".\Test1.cfg"

SEQUENCE-DATA=".\Test1.seq"

END-FILE-DEPENDENCIES

// Crozzle Size. The number of rows and columns.

// This crozzle will have 10 rows and 15 columns.

CROZZLE-SIZE

SIZE=10,15

END-CROZZLE-SIZE

// Horizontal Sequence Data.

HORIZONTAL-SEQUENCES

SEQUENCE=PETER,LOCATION=1,1

SEQUENCE=RONALD,LOCATION=1,10

SEQUENCE=MARK,LOCATION=3,1

SEQUENCE=GRAHAM,LOCATION=3,10

SEQUENCE=BRENDA,LOCATION=5,3

SEQUENCE=FRED,LOCATION=5,12

SEQUENCE=LARRY,LOCATION=7,9

SEQUENCE=BETTY,LOCATION=8,1

SEQUENCE=RON,LOCATION=9,7

SEQUENCE=SUSAN,LOCATION=9,11

SEQUENCE=MARY,LOCATION=10,4

END-HORIZONTAL-SEQUENCES

// Vertical Sequence Data.

VERTICAL-SEQUENCES

SEQUENCE=PAM,LOCATION=1,1

SEQUENCE=ROBERT,LOCATION=3,3

SEQUENCE=TOM,LOCATION=8,4

SEQUENCE=WENDY,LOCATION=4,5

SEQUENCE=GARY,LOCATION=7,7

SEQUENCE=ALAN,LOCATION=6,9

SEQUENCE=ROGER,LOCATION=1,10

SEQUENCE=ROSE,LOCATION=7,11

SEQUENCE=HARRY,LOCATION=3,13

SEQUENCE=AL,LOCATION=9,14

END-VERTICAL-SEQUENCES

# Appendix D – Test2.cfg

// Log File Configurations.

LOGFILE

// The default log file name.

DEFAULT="log.txt"

END-LOGFILE

// Letter Sequences Configurations.

// Limits on the number of unique letter sequences in the sequecnes file.

SEQUENCES-IN-FILE

MINIMUM=5

MAXIMUM=500

END-SEQUENCES-IN-FILE

// Crozzle Output Configurations.

CROZZLE-OUTPUT

INVALID-CROZZLE-SCORE="INVALID CROZZLE"

UPPERCASE=true

STYLE="<style> table, td { border: 1px solid black; border-collapse: collapse; } td { width:24px; height:18px; text-align: center; } </style>"

BGCOLOUR-EMPTY-TD=#777

BGCOLOUR-NON-EMPTY-TD=#ffffff

END-CROZZLE-OUTPUT

// Crozzle Configurations.

// Limits on the size of the crozzle grid.

CROZZLE-SIZE

MINIMUM-ROWS=5

MAXIMUM-ROWS=500

MINIMUM-COLUMNS=5

MAXIMUM-COLUMNS=500

END-CROZZLE-SIZE

// Limits on the number of horizontal letter sequences and

// vertical letter sequences in a crozzle.

SEQUENCES-IN-CROZZLE

MINIMUM-HORIZONTAL=10

MAXIMUM-HORIZONTAL=200

MINIMUM-VERTICAL=1

MAXIMUM-VERTICAL=5

END-SEQUENCES-IN-CROZZLE

// Limits on the number of

// intersecting vertical sequences for each horizontal sequences, and

// intersecting horizontal sequences for each vertical sequences.

INTERSECTIONS-IN-SEQUENCES

MINIMUM-HORIZONTAL=1

MAXIMUM-HORIZONTAL=500

MINIMUM-VERTICAL=2

MAXIMUM-VERTICAL=500

END-INTERSECTIONS-IN-SEQUENCES

// Limits on duplicate letter sequences in the crozzle.

DUPLICATE-SEQUENCES

MINIMUM=0

MAXIMUM=1

END-DUPLICATE-SEQUENCES

// Limits on the number of valid groups.

VALID-GROUPS

MINIMUM=1

MAXIMUM=2

END-VALID-GROUPS

// Scoring Configurations

// Points per letter that is at the intersection of

// a horizontal and vertical sequence within the crozzle.

INTERSECTING-POINTS

A=1

B=2

C=2

D=2

E=1

F=2

G=2

H=2

I=1

J=4

K=4

L=4

M=4

N=4

O=1

P=8

Q=8

R=8

S=8

T=8

U=1

V=16

W=16

X=32

Y=32

Z=64

z=64

END-INTERSECTING-POINTS

// Points per letter that is not at the intersection of

// a horizontal and vertical sequence within the crozzle.

NON-INTERSECTING-POINTS

A=0

B=0

C=0

D=0

E=0

F=0

G=0

H=0

I=0

J=0

K=0

L=0

M=0

N=0

O=0

P=0

Q=0

R=0

S=0

T=0

U=0

V=0

W=0

X=0

Y=0

Z=0

END-NON-INTERSECTING-POINTS

# Appendix E – Test2.seq

"[a-zA-Z]{2,}",310,153,11426,11889

AL,10,2,141,153

ALAN,10,4,284,298

ANGELA,10,6,424,440

BETTY,10,5,392,407

BILL,10,4,291,305

BRENDA,10,6,428,444

CHARLES,10,7,514,531

FRED,10,4,289,303

GARY,10,4,307,321

GEORGE,10,6,441,457

GRAHAM,10,6,432,448

HARRY,10,5,390,405

JACK,10,4,281,295

JESSICA,10,7,514,531

JILL,10,4,299,313

JOHNATHON,10,9,603,622

LARRY,10,5,394,409

MARK,10,4,299,313

MARY,10,4,313,327

MATTHEW,10,7,538,555

OSCAR,10,5,376,391

PAM,10,3,222,235

PETER,10,5,384,399

ROBERT,10,6,462,478

ROGER,10,5,383,398

RON,10,3,239,252

RONALD,10,6,448,464

ROSE,10,4,313,327

SUSAN,10,5,394,409

TOM,10,3,240,253

WENDY,10,5,391,406

# Appendix F – Test2.czl

// File dependencies.

FILE-DEPENDENCIES

CONFIG-DATA=".\Test2.cfg"

SEQUENCE-DATA=".\Test2.seq"

END-FILE-DEPENDENCIES

// Crozzle Size. The number of rows and columns.

// This crozzle will have 10 rows and 15 columns.

CROZZLE-SIZE

SIZE=1,15

END-CROZZLE-SIZE

// Horizontal Sequence Data.

HORIZONTAL-SEQUENCES

SEQUENCE=ROBERT,LOCATION=1,2

SEQUENCE=OSCAR,LOCATION=1,11

SEQUENCE=AL,LOCATION=2,8

SEQUENCE=JILL,LOCATION=3,2

SEQUENCE=AL,LOCATION=3,7

SEQUENCE=GEORGE,LOCATION=4,10

SEQUENCE=MARY,LOCATION=6,4

SEQUENCE=ROSE,LOCATION=6,10

SEQUENCE=RON,LOCATION=8,4

SEQUENCE=RON,LOCATION=8,8

SEQUENCE=JACK,LOCATION=9,1

SEQUENCE=FRED,LOCATION=9,12

SEQUENCE=ANGELA,LOCATION=10,6

END-HORIZONTAL-SEQUENCES

// Vertical Sequence Data.

VERTICAL-SEQUENCES

SEQUENCE=JESSICA,LOCATION=3,2

SEQUENCE=BILL,LOCATION=1,4

SEQUENCE=MARK,LOCATION=6,4

SEQUENCE=BRENDA,LOCATION=5,6

SEQUENCE=ALAN,LOCATION=2,8

SEQUENCE=ROSE,LOCATION=7,9

SEQUENCE=CHARLES,LOCATION=1,13

SEQUENCE=ROSE,LOCATION=1,15

SEQUENCE=WENDY,LOCATION=6,15

END-VERTICAL-SEQUENCES

# Appendix G – Test3.cfg

// Log File Configurations.

LOGFILE

// The default log file name.

DEFAULT=""

END-LOGFILE

// Letter Sequences Configurations.

// Limits on the number of unique letter sequences in the sequecnes file.

SEQUENCES-IN-FILE

MINIMUM=10

MAXIMUM=1

END-SEQUENCES-IN-FILE

// Crozzle Output Configurations.

CROZZLE-OUTPUT

INVALID-CROZZLE-SCORE="INVALID CROZZLE"

UPPERCASE=uppercase

STYLE="<style> table, td { border: 1px solid black; border-collapse: collapse; } td { width:24px; height:18px; text-align: center; } </style>"

BGCOLOUR-EMPTY-TD=777777

BGCOLOUR-NON-EMPTY-TD=#

END-CROZZLE-OUTPUT

// Crozzle Configurations.

// Limits on the size of the crozzle grid.

CROZZLESIZE

MINIMUM-ROWS=4

MAXIMUM-ROWS=400

MINIMUM-COLUMNS=4

MAXIMUM-COLUMNS=400

END-CROZZLESIZE

// Limits on the number of horizontal letter sequences and

// vertical letter sequences in a crozzle.

SEQUENCES-IN-CROZZLE

MINIMUM-HORIZONTAL=2

MAXIMUM-HORIZONTAL=1

MINIMUM-VERTICAL=2

MAXIMUM-VERTICAL=100

END-SEQUENCES-IN-CROZZLE

// Limits on the number of

// intersecting vertical sequences for each horizontal sequences, and

// intersecting horizontal sequences for each vertical sequences.

INTERSECTIONS-IN-SEQUENCES

MINIMUM-HORIZONTAL=1

MAXIMUM-HORIZONTAL=100

MINIMUM-VERTICAL=2

MAXIMUM-VERTICAL=1

END-INTERSECTIONS-IN-SEQUENCES

// Limits on duplicate letter sequences in the crozzle.

DUPLICATE-SEQUENCES

MIN=0

MAX=0

END-DUPLICATE-SEQUENCES

// Limits on the number of valid groups.

VALID-GROUPS

MAXIMUM=1

MINIMUM=2

END-VALID-GROUPS

// Scoring Configurations

// Points per letter that is at the intersection of

// a horizontal and vertical sequence within the crozzle.

INTERSECTING-POINTS

AAA=1

B=

C,2

D=XXXX

E=1

F=2

G=2

H=2

I=1

J=4

K=4

L=4

M=4

N=4

O=1

P=8

Q=8

R=8

S=8

T=8

U=1

V=16

W=16

X=32

Y=32

Z=64

END-INTERSECTING-POINTS

// Points per letter that is not at the intersection of

// a horizontal and vertical sequence within the crozzle.

NON-INTERSECTING-POINTS

A=0

B=0

C=0

D=0

E=0

F=0

G=0

H=0

I=0

J=0

K=0

L=0

M=0

N=0

O=0

P=0

Q=0

R=0

S=0

T=0

U=0

V=0

W=0

X=0

Y=0

Z=0

END-NON-INTERSECTING-POINTS

# Appendix H – Test3.seq

"[a-zA-Z]{2,}",310,146,10960,11416

AL,10,2,141,153

AL,10,2,141,153

ANGELA\*,10,7,424,441

,10,0,0,10

\*\*\*,10,3,126,139

SIT323,10,6,392,408

Charles,10,7,706,723

FRED,10,4,289,303

GARY,10,4,307,321

GEORGE,10,0,441,457

GRAHAM,10,6,444,448

HARRY,10,5,390,405

JACK,10,4,281,295

JESSICA,10,7,514,531

JILL,5,4,299,313

JOHNATHON,15,9,681,700

LARRY,10,5,394,409

MARK,10,4,299,313

MARY,10,4,313,327

MATTHEW,10,7,538,555

OSCAR,10,5,376,391

PAM,10,3,222,235

PETER,10,5,384,399

ROBERT,10,6,462,478

ROGER,10,5,383,398

RON,10,3,239,252

RONALD,10,6,448,464

ROSE,10,4,313,327

SUSAN,10,5,394,409

TOM,10,3,240,253

WENDY,10,5,391,406

# Appendix I – Test3.czl

// File dependencies.

FILE-DEPENDENCIES

CONFIG-DATA=".\Test3.cfg"

SEQUENCE-DATA=".\Test3.seq"

END-FILE-DEPENDENCIES

// Crozzle Size. The number of rows and columns.

// This crozzle will have 10 rows and 15 columns.

CROZZLE-SIZE

SIZE=10,y

END-CROZZLE-SIZE

// Horizontal Sequence Data.

HORIZONTAL-SEQUENCES

SEQUENCE=ROBERT,LOCATION=aaa,2

SEQUENCE=OSCAR,LOCATION=2,bbb

SEQUENCE=,LOCATION=3,2

SEQUENCE=MARY,6,4

LOCATION=6,11

SEQUENCE=GARY,LOCATION=8

SEQUENCE=?A?K?E,LOCATION=9,1

SEQUENCE=AL,LOCATION=9,14

END-HORIZONTAL-SEQUENCES

// Vertical Sequence Data.

VERTICAL-SEQUENCES

SEQUENCE=JESSICA,LOCATION=3,2

SEQUENCE=BILL,LOCATION=1.5,4

SEQUENCE=MARK,LOCATION=6,4

SEQUENCE=ROGER,LOCATION=6,6

SEQUENCE=HARRY,LOCATION=4,9

SEQUENCE=CHARLES,LOCATION=2,11

SEQUENCE=WENDY,LOCATION=2,15

END-VERTICAL-SEQUENCES