SIT323 Practical Software Development, Trimester 2, 2018

Assessment Task 1 – Validation and Testing

Due Date

Monday 9:00 am, August 27, 2018

Introduction

Assessment Tasks 1 and 2 comprise parts of the one project that must be developed using C# and MS Visual Studio. These require you to design, develop and test software related to the word game called crozzle (see class notes of week 1).

However, in order to free ourselves from words of a particular natural language and be generic, letter sequences will be used instead of words. This will permit values such as HELLO, BONJOUR, NIHAO, ONE, TWO, THREE, UN, DEUZ, TROIS, YI, ER, SAN, and sequences such as ABC, XML, HTTP, XXXX, REBMECED.

1. In brief, Assessment Task 1 focuses on updating an old program and testing that updated program. This old program is able to:
    - load data from three files related to configuration, a completed crozzle, and allowed letter sequences,
    - validate/invalidate the three input data files
    - validate/invalidate a crozzle that is described in these files,
    - compute the score of a crozzle, and
    - display that crozzle and its score.

    You will be provided with this old working program, valid input data files that this old program can read, and invalid input files that this old program can read and report issues. For each of these old input data files, you will be given a description of their format.

    This old program, old files and format descriptions can be found in the ZIP file called **Old Program and Old Files.zip**.

2. In brief, Assessment Task 2 requires you to focus on:
    - designing, developing and testing a cloud solution
    - loading data from three files related to creating a new crozzle such that its score is valid and is the highest possible that you can achieve,
    - ensuring that this new crozzle is valid,
    - saving this new crozzle to a file,
    - optimisation techniques,
    - appropriate scoping of code elements,
    - method characteristics such as cohesion and coupling,
    - the usage of return statements,
    - implementing exception handling, and
    - creating portfolio documents.

Objectives

Your main objectives for Assessment Task 1 are:

1. Update the old program to work with a different set of input data files that have new formats and extensions. These new formats are described in the following sections:
    - Crozzle file format (.czl)
    - Configuration file format (.cfg)
    - Letter Sequences file format (.seq)
2. Design and develop many unit tests.
3. Code to conventions and standards.
4. Create two portfolio documents.

Testing Requirements – Unit Tests

Your software solution for Assessment Task 1 requires testing. You must design and develop several unit tests for each of the following public methods.

1. Validator.IsBoolean
2. Validator.IsInt32
3. Validator.IsHexColourCode
4. KeyValue.TryParse
5. Crozzle.Score
6. CrozzleSequences,CheckDuplicateWords
7. Crozzle.Validate
8. Crozzle.ToStringHTML
9. CrozzleMap.GroupCount
10. Configuration.TryParse
11. WordList.TryParse
12. Crozzle.TryParse

Crozzle file format (.czl)

Your software solution for Assessment Task 1 must be able to read data from a crozzle CZL text file (see “Test1.czl” as an example).

1. Lines containing zero or more white spaces only are allowed.

2. Lines containing a comment or data are permitted to commence with 0 or more white spaces.

3. A line containing a comment is allowed. The symbol // will indicate the start of a comment, the end of line will represent the end of a comment. Some valid comment lines are as follows.

```
// This is valid.
// Creation date: 31/12/2018
    // Leading white spaces are valid too.
```

   Mixing data and a comment on one line is not allowed. For example, the following line is invalid:

```
SIZE=10,15        // A crozzle of 10 rows.
```

4. There will be a section of file dependencies that will commence with the keyword `FILE-DEPENDENCIES` and ends with the keyword `END-FILE-DEPENDENCIES`. Within this section:
   (a) There will be a line containing the file name of a configuration file. This can be an absolute or a relative file name. Two examples are below. The first is absolute, the second is relative.
   ```
   CONFIG-DATA="C:\\temp\config.txt"
   CONFIG-DATA=".\..\config.txt"
   ```

   (b) There will be a line containing the file name of a file that contains data related to letter sequences. This file name can also be absolute or relative. Two examples are:
   ```
   SEQUENCE-DATA="C:\\temp\sequences.txt"
   SEQUENCE-DATA=".\..\sequences.txt"
   ```

5. There will be a section of crozzle size data, i.e., the size of the 2D grid, which commences with the keyword `CROZZLE-SIZE` and ends with the keyword `END-CROZZLE-SIZE`.

   The first row of the crozzle is indexed by the number 1 (not 0), and the first column is indexed by the number 1 (not 0).

   Within this `CROZZLE-SIZE` section, there will be a line containing two positive integers. The first represents the actual number of rows in the crozzle, and the second represents the actual number of columns in the crozzle. For example, the following indicates a crozzle of size 15 rows and 20 columns.
   ```
   SIZE=15,20
   ```

6. There will be a section of data for each horizontal letter sequence in the crozzle. It commences and ends with the keywords `HORIZONTAL-SEQUENCES` and `END-HORIZONTAL-SEQUENCES`.

   Within this `HORIZONTAL-SEQUENCES` section, there is a line of data for each horizontal letter sequence displayed in a crozzle (as depicted in Figure 1). For example:
   ```
   SEQUENCE=OSCAR,LOCATION=2,9
   ```

   Such a line includes data related to a letter sequence, a comma, and the sequence location. It will contain:
   (a) the `SEQUENCE` keyword, equals symbol, and the actual letter sequence.
   (b) the `LOCATION` keyword, equals symbol, the row number, a comma, and the column number. The row number indicates which row contains all of the consecutive letters. The column number indicates which column the first letter is placed.

	R	O	B	E	R	T								
			I					O	S	C	A	R		W
	J	I	L	L						H				E
	E		L					H		A				N
	S							A		R				D
	S		M	A	R	Y		R		L	A	R	R	Y
	I		A		O			R		E				
	C		R		G	A	R	Y		S				
J	A	C	K		E									
					R									

Figure 1.

7. There will be a section of data for each vertical letter sequence in the crozzle. It commences and ends with the keywords `VERTICAL-SEQUENCES` and `END-VERTICAL-SEQUENCES`.

   Within this `VERTICAL-SEQUENCES` section, there is a line of data for each vertical letter sequence displayed in a crozzle (as depicted in Figure 1). For example:

   ```
   SEQUENCE=CHARLES,LOCATION=2,11
   ```

   Such a line includes data related to a letter sequence, a comma, and the sequence location. It will contain:

   (a) the `SEQUENCE` keyword, equals symbol, and the actual letter sequence.
   (b) the `LOCATION` keyword, equals symbol, the row number, a comma, and the column number. The row number indicates which row the first letter is placed. The column number indicates which column contains all of the consecutive letters.

   As there is no predefined ordering to these sections and section lines, the following 2 crozzle file examples are valid.

```
FILE-DEPENDENCIES
   CONFIG-DATA=".\Config.txt"
   SEQUENCE-DATA=".\Sequences.txt"
END-FILE-DEPENDENCIES

CROZZLE-SIZE
   SIZE=10,15
END-CROZZLE-SIZE

HORIZONTAL-SEQUENCES
   SEQUENCE=ROBERT,LOCATION=1,2
   SEQUENCE=JILL,LOCATION=3,2
END-HORIZONTAL-SEQUENCES

VERTICAL-SEQUENCES
   SEQUENCE=JESSICA,LOCATION=3,2
   SEQUENCE=BILL,LOCATION=1,4
END-VERTICAL-SEQUENCES
```

Crozzle file example 1.

```
CROZZLE-SIZE
   SIZE=10,15
END-CROZZLE-SIZE

FILE-DEPENDENCIES
   SEQUENCE-DATA=".\Sequences.txt"
   CONFIG-DATA=".\Config.txt"
END-FILE-DEPENDENCIES

VERTICAL-SEQUENCES
   SEQUENCE=BILL,LOCATION=1,4
   SEQUENCE=JESSICA,LOCATION=3,2
END-VERTICAL-SEQUENCES

HORIZONTAL-SEQUENCES
   SEQUENCE=JILL,LOCATION=3,2
   SEQUENCE=ROBERT,LOCATION=1,2
END-HORIZONTAL-SEQUENCES
```

Crozzle file example 2.

Configuration file format (.cfg)

Your software solution for Assessment Task 1 must be able to read data from a configuration CFG text file (an example can be seen in “Test1.cfg”):

1. Lines containing zero or more white spaces only are allowed.

2. Lines containing a comment or data are permitted to commence with 0 or more white spaces.

3. A line containing a comment is allowed. The symbol // will indicate the start of a comment, the end of line will represent the end of a comment. Some valid comment lines are as follows.

```
// This is valid.
// Creation date: 31/12/2018
    // Leading white spaces are valid too.
```

Mixing data and a comment on one line is not allowed. For example, the following line is invalid:

```
DEFAULT="log.txt"    // The default name of the log file.
```

4. There will be a section containing a file name. It will commence with the keyword `LOGFILE` and end with the keyword `END-LOGFILE`.

Within this section there will be a line containing the default name of a log file. This can be an absolute or a relative file name. For example:

```
DEFAULT="log.txt"
```

5. There will be a section of limits for the number of unique sequences allowed in the file of sequences. This section commences with the keyword `SEQUENCES-IN-FILE` and ends with the keyword `END-SEQUENCES-IN-FILE`.

Within this section there will be two lines, each containing a positive integer representing limits to the number of unique words in the sequences file. For example:

```
MINIMUM=10
MAXIMUM=1000
```

A valid sequences file does not contain duplicates.

6. There will be a section containing data related to the rendering of a crozzle. This section commences with the keyword `CROZZLE-OUTPUT` and ends with the keyword `END-CROZZLE-OUTPUT`.

   (a) There will be a line containing a string representing the score of an invalid crozzle. For example:

   ```
   INVALID-CROZZLE-SCORE="INVALID CROZZLE"
   ```

   (b) There will be a line containing a Boolean representing whether or not to display the crozzle using uppercase or lowercase letters. For example:

   ```
   UPPERCASE=true
   ```

   (c) There will be a line containing a string representing a HTML style, which can be used to manage how to render a crozzle. For example:

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

   (d) There will be two lines containing colour codes representing the background colour of empty and non-empty crozzle cells. For example:

   ```
   BGCOLOUR-EMPTY-TD=#777777
   BGCOLOUR-NON-EMPTY-TD=#ffffff
   ```

7. There will be a section containing data related to the size of a crozzle. This section commences with the keyword `CROZZLE-SIZE` and ends with `END-CROZZLE-SIZE`.

   Within this section there will be four lines containing such limit values. For example:

   ```
   MINIMUM-ROWS=4
   MAXIMUM-ROWS=400
   MINIMUM-COLUMNS=8
   MAXIMUM-COLUMNS=800
   ```

8. There will be a section containing limits on the number of horizontal and vertical sequences permitted in a valid crozzle. This section commences with the keyword `SEQUENCES-IN-CROZZLE` and ends with `END-SEQUENCES-IN-CROZZLE`.

   Within this section there will be four lines containing such limit values. For example:

   ```
   MINIMUM-HORIZONTAL=1
   MAXIMUM-HORIZONTAL=100
   MINIMUM-VERTICAL=1
   MAXIMUM-VERTICAL=100
   ```

9. There will be a section containing limits on the number of intersecting vertical sequences for each horizontal sequence, and the number of intersecting horizontal sequences for each vertical sequence. This section commences with the keyword `INTERSECTIONS-IN-SEQUENCES` and ends with `END-INTERSECTIONS-IN-SEQUENCES`.

   Within this section there will be four lines containing such limits. For example:

   ```
   MINIMUM-HORIZONTAL=1
   MAXIMUM-HORIZONTAL=100
   MINIMUM-VERTICAL=1
   MAXIMUM-VERTICAL=100
   ```

10. There will be a section containing limits on the number of duplicate sequences allowed within a crozzle. This section commences with the keyword `DUPLICATE-SEQUENCES` and ends with `END-DUPLICATE-SEQUENCES`.

    Within this section there will be two lines containing such limits. For example:

    ```
    MINIMUM=1
    MAXIMUM=1
    ```

11. There will be a section containing limits on the number of valid groups allowed within a crozzle. This section commences with the keyword `VALID-GROUPS` and ends with `END-VALID-GROUPS`.

    Within this section there will be two lines containing limits on the number of valid disconnected groups of sequences that are allowed within a crozzle. For example:

    ```
    MINIMUM=1
    MAXIMUM=1
    ```

    A maximum value of 1 indicates that all sequences in the crozzle are within 1 connected group of sequences. The crozzle in Figure 1 shows 2 disconnected groups of sequences; one group is shaded yellow, the other group is shaded orange.

12. There will be a section containing the number of points per letter, where each such letter is at the intersection of a horizontal and vertical sequence. This section commences with the keyword `INTERSECTING-POINTS` and ends with `END-INTERSECTING-POINTS`.

    Within this section there will be 26 lines, one for each letter of the alphabet. Each such line contains a letter, an equals symbol, and points for that intersecting letter. For example:

    ```
    A=1
    B=2
    C=2
    …
    Z=64
    ```

    The two letters B and C are the intersecting letters on the 1st and 2nd rows of Figure 1. These letters would score 2+2=4 points, based on this example.

13. There will be a section containing the number of points per letter, where each such letter is *not at the intersection* of a horizontal and vertical sequence. This section commences with the keyword `NON-INTERSECTING-POINTS` and ends with `END-NON-INTERSECTING-POINTS`.

    Within this section there will be 26 lines, one for each letter of the alphabet. Each such line contains a letter, an equals symbol, and points for that letter. For example, assume each letter is worth 0 points such as:

    ```
    A=0
    B=0
    C=0
    …
    Z=0
    ```

    The seven letters R, O, E, R and T are the non-intersecting letters on the 1st row of Figure 1. These letters would score 0 points, based on this example.

Letter Sequences file format (.seq)

Your software solution for Assessment Task 1 must be able to read comma separated values from a sequence SEQ text file (see “Test1.seq” as an example).

1. A valid sequence file does not contain duplicate letter sequences.

2. A valid sequence file contains a header followed by 0 or more lines of data related to sequences, one line per letter sequence. For example:

    ```
    [A-Z]+,40,14,1042,1096
    APPLE,10,5,370,385
    BAG,10,3,202,215
    GO,10,2,150,162
    ZERO,10,4,320,334
    ```

(a) The header contains:
   i. a regular expression pattern for specifying allowed characters in a letter sequence
   ii. a batch total for the 2nd field of sequence data
   iii. a batch total for the 3rd field of sequence data
   iv. a batch total for the 4th field of sequence data
   v. a batch total for the 5th field of sequence data

(b) A line of sequence data contains 5 fields separated by commas:
   i. the letter sequence
   ii. the number of points obtained when using this sequence in a valid crozzle
   iii. a length check, for checking the number of characters in the sequence
   iv. a consistency check, the sum of ASCII values from each character in the sequence
   v. a hash total, the sum of fields 2, 3 and 4

Validating files and crozzle

1. Your software solution for Assessment Task 1 must be able to validate the three text input data files (czl, cfg and seq).

   Invalid aspects of these files are written to a log file and available via the GUI.

2. Your software solution for Assessment Task 1 must be able to validate the crozzle too, but only if the input data files are valid. Consequently, it is possible to have perfectly valid input files, but the crozzle is invalid.

   For example, if the configuration file contained the following limits, the crozzle in Figure 1 is invalid. It would be invalid because the maximum number of groups permitted is 1 but that crozzle in Figure 1 contains 2 groups; the yellow group and the orange group.

```
VALID-GROUPS
  MINIMUM=1
  MAXIMUM=1
END-VALID-GROUPS
```

   Invalid aspects of a crozzle are written to a log file and available via the GUI.

Displaying a crozzle and its score

1. Whether or not the input files and crozzle are valid, the existing program attempts to display a valid or invalid crozzle on the GUI. For example, depending on configuration settings or the letter sequences, the crozzle in Figure 1 might be valid/invalid, but it can still be displayed.

2. The existing program computes and displays the score of a valid crozzle on the GUI. However, an appropriate configuration value is displayed for an invalid crozzle.

Constraints

The old program considered and adhered to constraints when validating a crozzle and determining a crozzle score. These constraints are:

1. A valid crozzle must conform to configuration settings.
2. Each sequence of two or more horizontal (vertical) characters delimited by spaces or a crozzle edge must form a letter sequence that can be found in the file of sequences.
3. A horizontal letter sequence can only run from left to right.
4. A vertical letter sequence can only run from high to low.
5. Diagonal sequences of characters, which can be formed by horizontal and vertical letter sequences, are not pertinent to scoring or validity.

Latest data files

To help you design and develop your unit tests and update the existing program, you will be provided with 9 files that are based on the above formats. These file can be found in the ZIP file called **Assessment Task 1 Files.zip**. The validities of each file and the associated crozzle are presented in the following table.

1. The three files for Test1 are all valid, and also the crozzle is valid. Consequently, your program should not detect and report issues for this group of test files.
2. The three files for Test2 are all valid, but the crozzle is actually invalid. Your program must display as much of this invalid crozzle as possible.
3. The three files for Test3 are all invalid, and also the crozzle is invalid. Portions of this invalid crozzle cannot be displayed as, for example, there is not enough data in the crozzle file to identify the location of a word. However, other portions of this invalid crozzle can be displayed on the GUI.

Test	File Name	File Validity	Crozzle Validity
1	Test1.czl	valid	valid
	Test1.cfg	valid	
	Test1.seq	valid	
2	Test2.czl	valid	invalid
	Test2.cfg	valid	
	Test2.seq	valid	
3	Test3.czl	invalid	invalid
	Test3.cfg	invalid	
	Test3.seq	invalid	

Portfolio Documents

You must produce two Word documents of professional quality. In general, one document will present outputs of your updated program, and the other document presents the unit tests.

1. **Portfolio Document Program** must include the following:
   - a title page
   - a contents page
   - an introduction page
   - document body
   - an appendix

   The document body contains sections, one section for each test (Test1, Test2, and Test3). Such a section will include a brief summary and the outputs (such as GUI, error list, log file) of that particular test.

   The appendix contains sections too, one section for each test. Such a section will include copies of the input data files of a particular test.

2. **Portfolio Document Unit Tests** must include the following:
   - a title page
   - a contents page
   - an introduction page
   - document body
   - an appendix

   The document body contains sections, one section for each unit test. Such a section will include a brief summary and the results of that particular unit test.

   The appendix contains sections too, one section for each unit test. Such a section will include copies of the input data and files (if any) of a particular unit test.