**1305ENG Major Assignment**

**Slater Black (s5259373)**

Contents

Problem Statement 2

Requirements Section 2

User Requirements 2

Software Requirements 2

Software Design 3

Program I/O 3

Data Section 3

High Level Design 3

Functions Section 4

List of all functions in the software. 4

Sample Output 4

Test Data 5

Requirement Acceptance Tests 5

Detailed Software Testing 6

User Instructions 7

Appendix A – Source Code 8

# Problem Statement

**(In this section you write a few sentences about what needed to be done for the assigment)**

The goal is to create a program in Perl that accepts Windows file names from the DOS command line, validates the input using Perl regular expressions and then outputs information about each valid file name or an error message for invalid file names.

The program must accept multiple file names in a single command. Each file name can consist of path and file name with multiple levels in the path.

# Requirements Section

## User Requirements

**(In this section you detail how a user is supposed to interact with or use your program code. This is a general description of the *interface* of the program)**

The following outlines the user requirements for the program:

* The user shall be able to enter as many file names on the command line as they like.
* Each one can consist of the full path of the file with multiple levels (eg. *z:\path\filename.doc*) or simply the file name without the path (eg. f*ilename.doc*).
* The user shall also be able enter a directory name without a file name and receive valid output.
* The user should be able to enter as many levels for each file as they want.
* File and directory names can be from 1 to 64 characters long.
* File extensions should be 3 or 4 characters preceded by a dot (eg *filename*.doc or *filename*.html)

## Software Requirements

**(In this section you spell out what the requirements were, taken from the assigment handout. If you have added your own features, these should also be listed here.)**

The following outlines the software requirements for the program:

1. The program shall accept multiple file names as arguments from the command line.
2. The program shall display the details of all valid files .
3. The program shall display an appropriate message if a file does not exist or if a file name is invalid.
4. The program shall display a message if an argument is a directory instead of a file.
5. Each file name can be a simple file name or include the full path of the file with one or more levels.
6. All file names must start with an alphabetical character to be considered a valid file name.
7. Valid file name extensions must be 3 or 4 alphabetical characters preceded by a dot.
8. Directory/level names must start with an alphabetical character to be considered valid.
9. The program should be able to accept as many levels for each file name as the user wants to input. This is limited only by the number of levels allowed in Windows (approximately 120).
10. Each level of the file name can be a maximum of 64 characters.

# Software Design

## Program I/O

**List the inputs and outputs from the program. For example, if input comes from particular files, list these. If input is to come from the keyboard, show the form of this input.**

## Data Section

For each major data structure in the list the following information is provided:

1. Type of structure (integer, array etc),
2. Description of where and how it is used
3. List of data members, and what each one is for do
4. List of functions that use it

## High Level Design

**(In this section explain or draw a little block diagram of how your program works)**

Get User input

Print details:

File name, access mode, size, last accessed and last modified.

Process directory

Check if it is a file or directoy

Output Message:

Validate input (regular expr)

true file

false neither directory

Output Message:

## Functions Section

### List of all functions in the software.

For each function in the list the following information is provided:

1. a brief description of what it does (1 or 2 sentences);
2. a list of the input parameters, and their data types, and what they are used for;
3. a list of any side effects caused by the function (ie change global or member variables, changes data passed by reference from calling function etc)
4. a description of the function’s return value
5. A brief description of how the function works (overview of the algorithm).
6. How complete the function is – does it work properly, are there any errors, etc.

# Sample Output

**Show examples of your program in the form of screenshots or other outputs. Try to demonstrate as many features as possible without including too many images.**

# Test Data

## Requirement Acceptance Tests

| **Software  Requirement No** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| --- | --- | --- | --- | --- |
| 1 | User can generate a new game of custom dimensions and specify the number range allowed. Arithmetic operators must be randomly chosen, as well as the correct. | FULLY implemented | PASS | N/A |
| 2 | The user can ask for some valid numbers to be displayed. The game will randomly select a correct number to show the user. | FULLY implemented | PASS | N/A |
| 3 | The game board is to be displayed after every attempt with all the guessed values present. | FULLY implemented | PASS | N/A |
| 4 | User plays the game by entering and removing numbers from the grid. | FULLY implemented | PASS | N/A |
| 5 | The time taken, size of the grid, difficulty and number of times a user has removed a number from the grid should be taken into account calculating the players final score. | FULLY implemented | PASS | N/A |
| 6 | User scores will be displayed as the game progreses |  |  |  |
| 7 | Game saves and can be resumed later | FULLY implemented | PASS | N/A |
| 8 | Integrate a main menu that can view high scores from a file | FULLY implemented | PASS | N/A |
| 9 | Add the “%” operator (modulus) | FULLY implemented | Fail | Mathematical operations currently do not reach the correct answer |
| 10 | Allow the user to enter a value that will flag a solution and make it an invalid guess | FULLY implemented | PASS | N/A |
| 11 | Allow the user to ask for help obtain a value | FULLY implemented | PASS | N/A |
| 12 | Add another feature (explain in-----) |  |  |  |

## Detailed Software Testing

**(In this table you fill out details about what additional tests you have done in addition to the requirements acceptance tests to ensure that there are no bugs in your program)**

| **No** | **Test** | **Expected Results** | **Actual Results** |
| --- | --- | --- | --- |
| **1.0** | **Files** |  |  |
| 1.1  1.2  1.3  1.4 | File name with: - no directory or path  (*filename.doc*) - directory but no levels (*z:\filename.doc*) - one level in path (*z:\level\filename.doc*) - more than one level in path *(z:\level1\level2\filename.doc)* | For each case 1.1 - 1.4:  a. File exists: details of file displayed.  b. File doesn’t exist: “File does not exist” displayed | As expected |
| 1.5 | File name with non alphabetical character as first character | Display “Invalid file name” | As expected |
| 1.6 1.7 1.8 | File extension with: - 3 or 4 alphabetical characters - more than 4 characters  - less than 3 characters | For each case 1.6 - 1.9:  Display “Invalid file name” | “File does not exist” displayed |
| **2.0** | **Directories** |  |  |
| 2.1 2.2 2.3 | Directory with:  - no levels  - one level or  - more than one level | For each case 2.1 - 2.3:  a. Directory exists: “Is a directory!” displayed.  b. Directory doesn’t exist: “Directory does not exist” displayed | As expected |
| 2.4 | Directory with non alphabetical character as first character (eg 2:\) | Display “Invalid file name” | “File does not exist” displayed |
| 2.5 | Level of directory with non alphabetical character as first character (eg z:\2level) | Display “Invalid file name” | “File does not exist” displayed |

# User Instructions

**(Here you state what a user needs to do to run and use your program(s), this also includes and special instructures on how to compile the program(s) as necessary)**

* Open the file question1.cgi in OptiPerl.
* Run the program in console mode in OptiPerl.
* Enter one or more file and/or directory names as command line arguments. They can be existing or fictitious files.
* For existing files details of each file will be displayed.
* For invalid file names an error message will be displayed.
* For files that do not exist an error message will be displayed.

# Appendix A – Source Code

Include all source code here.