|  |
| --- |
|  |
| CS4218 Milestone 1 |
|  |

|  |
| --- |
|  |

**Team 05**

KOH CHENG GEE A0126199W

MUHAMMAD RAZALI A0133267H

PHOON QIANONN A0126232U

TANG WEI REN A0125531R

Contents

[1. Implementation for Basic Functionality (BF) 3](#_Toc475310854)

[1.1 Implementation for Shell: Calling applications 3](#_Toc475310855)

[1.2 Implementation for Shell: Quoting 3](#_Toc475310856)

[1.3 Implementation for Shell: Semicolon Operator 3](#_Toc475310857)

[1.4 Implementation for Applications: cat 3](#_Toc475310858)

[1.5 Implementation for Applications: cd 3](#_Toc475310859)

[1.6 Implementation for Applications: pwd 3](#_Toc475310860)

[1.7 Implementation for Applications: echo 3](#_Toc475310861)

[1.8 Implementation for Applications: head 3](#_Toc475310862)

[1.9 Implementation for Applications: tail 3](#_Toc475310863)

[2. Implementation for Extended Functionality 1 (EF1) 3](#_Toc475310864)

[2.1 Implementation for Shell: Globbing 3](#_Toc475310865)

[2.2 Implementation for Shell: IO-redirection 3](#_Toc475310866)

[2.3 Implementation for Applications: cal 3](#_Toc475310867)

[2.4 Implementation for Applications: grep 4](#_Toc475310868)

[2.5 Implementation for Applications: sort 4](#_Toc475310869)

[3. Unit Testing for Basic Functionality (BF) 4](#_Toc475310870)

[3.1 Unit Testing for Shell: Calling applications 4](#_Toc475310871)

[3.2 Unit Testing for Shell: Quoting 4](#_Toc475310872)

[3.3 Unit Testing for Shell: Semicolon Operator 4](#_Toc475310873)

[3.4 Unit Testing for Applications: cat 4](#_Toc475310874)

[3.5 Unit Testing for Applications: cd 4](#_Toc475310875)

[3.6 Unit Testing for Applications: pwd 5](#_Toc475310876)

[3.7 Unit Testing for Applications: echo 5](#_Toc475310877)

[3.8 Unit Testing for Applications: head 5](#_Toc475310878)

[3.9 Unit Testing for Applications: tail 5](#_Toc475310879)

[4. Unit Testing of Extended Functionality 1 (EF1) 5](#_Toc475310880)

[4.1 Unit Testing for Shell: Globbing 5](#_Toc475310881)

[4.2 Unit Testing for Shell: IO-redirection 5](#_Toc475310882)

[4.3 Unit Testing for Applications: cal 5](#_Toc475310883)

[4.4 Unit Testing for Applications: grep 6](#_Toc475310884)

[4.5 Unit Testing for Applications: sort 6](#_Toc475310885)

[5. Test Cases for Extended Functionality 2 (EF2) 6](#_Toc475310886)

[5.1 Test Cases for Shell: Pipe Operator 6](#_Toc475310887)

[5.2 Test Cases for Shell: Command Substitution 6](#_Toc475310888)

[5.3 Test Cases for Applications: date 6](#_Toc475310889)

[5.4 Test Cases for Applications: sed 6](#_Toc475310890)

[5.5 Test Cases for Applications: wc 7](#_Toc475310891)

# Implementation for Basic Functionality (BF)

## Implementation for Shell: Calling applications

* Plans for implementation

## Implementation for Shell: Quoting

* Plans for implementation

## Implementation for Shell: Semicolon Operator

* Plans for implementation

## Implementation for Applications: cat

* Plans for implementation

## Implementation for Applications: cd

The cd command changes the current working directory

|  |  |
| --- | --- |
| Command | Description |
| cd | change to home directory |
| cd ~ | Change to home directory |
| cd / | Change to the root directory  For Mac or Linux   * Linux uses “/” for its “file separator” * However, for Linux, if a user entered “\” as the “file separator” instead of “/” there will be no auto correction as “\” is a valid folder & file name in the Linux. An appropriate error message will be display if the “path” entered does not exist in the shell system. |
| cd / | Change to the root directory  For Windows   * Windows uses “\” for its “file separator”. * If “/” is mistakenly used as the “file separator” in the “path” entered, the shell system will auto correct “/” to “\”. * The following observation was made in the “command prompt” on Windows, hence this correction was implemented to help ease the user as their intention is clear. * As “/” is an invalid folder & file name in Windows, the auto-correction will not cause any issues |
| cd \ | For Windows:   * Windows uses “\” for its “file separator”. |
| cd . | No change in directory. Remain at current working directory |
| cd .. | Change to parent directory |
| cd [PATH] | Change the directory to the “PATH”. If the “PATH” does not exist it will return an error message   * Windows uses “\” for its “file separator”. * If the shell system is run on Windows, auto correction of the “file separator” from “/” to “\” will take if “/” is mistakenly used as the “file separator” in the “PATH” entered. * The following observation was made in the “command prompt” on Windows, hence this correction was implemented to help ease the user as their intention is clear. * As “/” is an invalid folder & file name in Windows, the auto-correction will not cause any issues * “PATH” is a “relative path directory” e.g. “folder1/folder2” * “PATH” as specific in the project document cannot be an “absolute path directory” e.g. /user/home/directory”. A “/” (for mac & Linux) and “\” (for Windows) indicates that the “PATH” is an “absolute path directory” hence the “shell system” will return an error message if “/” or “\” is detected as the first char of the “PATH” |

## Implementation for Applications: pwd

The pwd command prints the current working directory followed by a newline

|  |  |
| --- | --- |
| Command | Description |
| pwd | Display the current working directory followed by a newline   * Use “*Environment.currentDirectory*” to get the current working directory * Add “*System.lineSeparator*” to insert a newline after displaying the results |

## Implementation for Applications: echo

* Plans for implementation

## Implementation for Applications: head

* Plans for implementation

## Implementation for Applications: tail

* Plans for implementation

# Implementation for Extended Functionality 1 (EF1)

## Implementation for Shell: Globbing

* Plans for implementation

## Implementation for Shell: IO-redirection

* Plans for implementation

## Implementation for Applications: cal

* Plans for implementation

## Implementation for Applications: grep

* Plans for implementation

## Implementation for Applications: sort

* The “sort” commands sort a specific file or input in a sorted order
* The sorted order is defined in the order of “special character”, “numbers”, “capital letters” and “simple letters”
* If “-n” is specific after the “sort” command (e.g. sort [-n] [FILE]), the first word of a line will be treated as a number. The first word will be treated as number only if it contains all numeric characters (e.g 8 days later, so “8” is the first word). If the first word will be treated as a normal word if it contains other types of characters(e.g. 8days later, “8days” is the first word)
* If 2 or more “FILES” are provided, the lines in the “FILES” will be combined and sorted together

|  |
| --- |
| Methods & Description |
| public void run(String args[], InputStream stdin, OutputStream stdout)   * args[] takes in an array of String which contains either “-n” or “FILE” * stdin takes in inputStream * stdout takes in outputStream |
| public String sortStringSimple(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with simple letter |
| public String sortStringCapital(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with capital letter |
| public String sortNumbers(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with numbers |
| public String sortSpecialChars(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars |
| public String sortSimpleCapital(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with capital letter and simple letter. In the order of capital letter followed by simple letter |
| public String sortSimpleNumbers(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with numbers and simple letter. In the order of numbers followed by simple letter |
| public String sortSimpleSpecialChars (String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with capital letter and simple letter. In the order of capital letter followed by simple letter |
| public String sortCapitalNumbers(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with numbers and capital letter. In the order of numbers followed by captial letter |
| public String sortCaptialSpecialChars(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars and capital letter. In the order of special chars followed by capital letter. |
| public String sortNumbersSpecialChars(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars and numbers. In the order of special chars and numbers. |
| public String sortSimpleCapitalNumber(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with numbers, capital letter and simple letter. In the order of numbers, capital letter followed by simple letters. |
| public String sortSimpleCapitalSpecialChars(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars, capital letter and simple letter. In the order of special chars, capital letter followed by simple letters. |
| public String sortSimpleNumbersSpecialChars (String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars, numbers and simple letter. In the order of special chars, numbers followed by simple letters. |
| public String sortCaptialNumbersSpecialChars(String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars, numbers and capital letter. In the order of special chars, numbers followed by capital letters. |
| public String sortAll (String toSort)   * toSort takes in String. * Data from InputStream will have to be converted to “String” type first The conversion is done inside public void run(String args[], InputStream stdin, OutputStream stdout). * Sort and return a string of lines starting with special chars, numbers, capital letter, simple letters. In the order of special chars, numbers, capital letters followed by simple letters. |

# Unit Testing for Basic Functionality (BF)

## Unit Testing for Shell: Calling applications

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Shell: Quoting

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Shell: Semicolon Operator

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: cat

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: cd

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: pwd

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: echo

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: head

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: tail

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

# Unit Testing of Extended Functionality 1 (EF1)

## Unit Testing for Shell: Globbing

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Shell: IO-redirection

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: cal

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: grep

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Unit Testing for Applications: sort

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

# Test Cases for Extended Functionality 2 (EF2)

## Test Cases for Shell: Pipe Operator

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Test Cases for Shell: Command Substitution

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Test Cases for Applications: date

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Test Cases for Applications: sed

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests

## Test Cases for Applications: wc

* Summary of Test cases provided
* What have you covered during testing
* Did you have any plans for generation tests