ISAD1000 Introduction to Software Engineering – Exercise Submission

Name: Thomas Beyene
Student ID: 20213053
Practical Class: Friday 8-10am
Venue: Building 314, Room 218

Introduction:

For my assignment I have implemented the Category 1 and Category 2a. I have implemented Category 1 and 2 production code inside the python file ModularityConv, I added extra functionalities to the functions with user input for the Category 1 functionalities and input/outfile txt file for Category 2a. To test the test designs I have used black box testing for all functionalities and white box testing for Category 1a and 1c.

Modularity:

Once the program started, menu options are displayed, the user can make a selection from from option 1 to 5 for the functionalities. The menu has a range of functionalities that user can select from. It run through a loop that takes in the user's input and passes it into the function and returns the output and menu options again and provides the user to have anther opportunity to enter their input if they entered is invalid. To exit the program by entering 0.

The programs achieves high cohesion as the code that can be easily reused. This is shown in the removeNumericNConvert function reusing the removeNumeric() function. Some functionalities from each category will have many functions within it that it calls upon. This is to avoid poor cohesion within the code.

Use of parameters and return values within functions to avoid global variables. It results in an increase in coupling however there isn't a high number of parameters (max. 2) for each functions.

Checklist:
Are each functionality is defined as a separate function?
Yes - each functionality has their own function

Is the code well organised in terms of placement of components?
Yes

Are functions using return values?
Yes

Are there any variables that have not been initialised?
No

Are functions performing its task without any errors?

Yes

Is there any repeated and unneccessary code?
Yes in RemoveNumericAndConvertOption() - can call the convertString() function inside it instead of having repeated code.

Are there any global variables?
No

Refactoring was done for RemoveNumericAndConvertOption function. Firstly I created a function called RemoveNumericAndConvert which takes in variables and is called in the RemoveNumericAndConvertOption function. Additionally I modified the ConvertLengthOption(), by writing a function for the conversion of lengths called ConvertLength() and separating the write and read operations into the ConvertLengthOption where it can call the ConvertLength() function.

Module description:

convertString(INPUT, upper_lower)
This function takes in an INPUT and an option of ‘U’ or ‘L’
The input string is converted by the option ‘U’ or ‘L’, which converts the string to uppercase or lowercase respectively
The module returns the result OUTPUT, which displays the converted string.

convertStringOption()
The functions takes in user input, which firsts asks the user want to convert the string uppercase or lowercase, then it prompts the users to enter a string to convert. It then run the convertString function. If a string that is entered is invalid then the program is unable to convert the string.

ContainsNumbericValue(INPUT)
The function takes in an inputString which checks if it contains numeric values from the range of 0 to 9 which is stored in a list. It then return a Boolean value of True if the string contains a numeric value, otherwise it would return a False statement

ContainsNumbericValueOption()
The function takes in user input, which firsts prompts the user to enter a string, then performs the ContainsNumbericValue() function. If the string entered contains numeric values then it prints the statement "The string contains numeric value." If it doesn’t then it prints the statement "The string does not contain numeric value."

IsValidNumber(INPUT)
The function check if all characters in INPUT contain numeric values of the range 0 to 9 which is stored in a list. It then return a Boolean value of True if the string contains a numeric value, otherwise it would return a False statement.

IsValidNumberOption()
The function takes in user input, which firsts prompts the user to enter a string, then performs the ContainsNumbericValue() function. If the string entered contains numeric values then it prints the statement "The string is the valid." If it doesn't then it prints the statement "The string is not valid."

RemoveNumeric(INPUT)
This functions loops through each character in the string and checks if there are digits. If it is not a number than the string is added to the output string. The result output string will be the input string without numbers. The return value is string after removing any numeric values

RemoveNumericAndConvert(INPUT, upper_lower)
This functions loops through each character in the string and checks if there are digits. If it is not a number than the string is added to the output string. The result output string will be the input string without numbers. It then prompts to convert the string uppercase or lowercase. The return value is string after removing any numeric values, also converted to either uppercase or lowercase based on the user choice.

RemoveNumericAndConvertOption()
The function takes in user input, which firsts prompts the user to enter a string, the string goes through the RemoveNumeric(inString) function, that checks if there are digits and remove them. It then prompt user for input to convert the newly modified string to be uppercase or lowercase.

ConvertLength(length, measurement):

This module takes in a float value for the length and an integer from the range of 1 to 4 that represents the length measurement. The length measurements are represented by 1 being metres will be converted to feet, 2 being feet will be converted to metres, 3 being centimetres will be converted to inches and 4 being inches will be converted to centimetres.
It then passes through the conversion into the output variable and returns output.

ConvertLengthOption():
This function takes in an input file that contains a that represents a length value as a float and operates on each line of the file. For each line, the length values are converted firstly from metres are converted to feet, then feet to metres, centimetres to inches and inches to centimetres. These values are written into the output file.

Each function will ask the user whether or not they'd like to input data in the form of a string through their keyboard or through the reading of a data file.

Category 1: For Category 1 the functionalities performs string conversions.

Converting a given string to upper case or lower case.
The functionality operates by taking in a string and converting the letter to either uppercase or lowercase.
The functionality should only accept letters and ignore special characters and numbers

Identify whether a given string is a valid number or not. - The functionality operates by taking in a string and checking if the string only contain a number between 0 to 9.
The program will then display a message of whether the string is valid or the string is not valid

Identify whether numeric values are in a given string.
The functionality operates by taking and check if there are any numbers within the string
the program will display a message of whether there was or wasn't a number within the string

Remove any numeric values in a given string and then convert the string to upper case or lower Case. The functionality operates by taking in a string and remove any numbers within it, after performing that operation it will either convert the string to upper case or lower case.

In Category 2, the functionalities performs conversion of measurements:
Converting a number which represents a length given in meters to feet and vice versa and centimetres to inches and vice versa.
The functionality operates by taking in an input file in which the conversions meters to feet, feet to meters, centimetres to inches and inches to centimetres are done

Testing:

Converting a given string to upper case or lower case.

Equivalence Partitioning:

Submodule: convertString()
Imports: INPUT(str), upper_lower(str)
Exports: OUTPUT (str)

"Identify whether numeric values are in a given input parameter."

Category	Test Data	Expected Result
String: Last four digits of student ID	INPUT = “3053” upper_lower = “L”	OUTPUT = “3053”
String: Last name	INPUT = “Beyene” upper_lower = “U”	OUTPUT =“BEYENE”
String: Full name as per student ID card	INPUT = “Thomas Beyene” upper_lower = “U”	OUTPUT =“THOMAS BEYENE”
String: Movie on watchlist	INPUT = “Interstellar” upper_lower = “L”	OUTPUT = “interstellar”
String: Movie on watchlist	INPUT = “Interstellar” upper_lower = “O”	OUTPUT = “String cannot be converted”

Submodule: ContainsNumbericValue()
Imports: INPUT(str)
Exports: numbers(bool)

Identify whether numeric values are in a given string.

Category	Test Data	Expected Result
String: Last four digits of student ID	INPUT = “3053”	numbers = True
String: Last name	INPUT = “Beyene”	numbers = False
String: Full name as per student ID card	INPUT = “Thomas Beyene”	numbers = False
String: Movie on watchlist	INPUT = “Interstellar”	numbers = False

Submodule: RemoveNumeric()
Imports: INPUT(str)
Exports: OUTPUT(str)

Remove any numeric values in a given string

Category	Test Data	Expected Result
String: Last four digits of student ID	INPUT = “3053”	Output = “”
String: Last name	INPUT = “Beyene”	Output = “Beyene”
String: Full name as per student ID card	INPUT = “Thomas Beyene”	Output = “Thomas Beyene”
String: Movie on watchlist	INPUT = “Interstellar”	Output = “Interstellar”

Submodule: IsValidNumber()
Imports: INPUT(str)
Exports: numbers(bool)

Identify whether numeric values are in a given string.

Category	Test Data	Expected Result
String: Last four digits of student ID	INPUT = “3053”	ValidNumber = True
String: Last name	INPUT = “Beyene”	ValidNumber = False
String: Full name as per student ID card	INPUT = “Thomas Beyene”	ValidNumber = False
String: Movie on watchlist	INPUT = “Interstellar”	ValidNumber = False

Submodule: RemoveNumericAndConvert()
Imports: INPUT(str), upper_lower(str)
Exports: OUTPUT(str)

Remove any numeric values in a given string

Category	Test Data	Expected Result
String: Last four digits of student ID	INPUT = “3053” upper_lower = U	OUTPUT = “ ”(empty string)
String: Last name	INPUT = “Beyene” upper_lower = L	OUTPUT = “beyene”
String: Full name as per student ID card	INPUT = “Thomas Beyene” upper_lower = L	OUTPUT = = “thomas beyene”
String: Movie on watchlist	INPUT = “Interstellar” upper_lower = U	OUTPUT = “INTERSTELLAR”

Boundary Value Analysis:

Submodule: ConvertLength()
Imports: length(float), measurement(str)
Exports: output(float)

Converting a number which represents a length given in meters to feet and vice versa and centimetres to inches and vice versa – Boundaries between the different measurement conversions.

Category	Test Data	Expected result
Invalid/0	Length = 100 metres measurement = -1/0	output = “Invalid measurement”, 328.08 feet
measurement = 0/10	Length = 100 metres/feet measurement = 9/10	output = 328.08 feet, 30.48 metres
Measurement = 10/20	Length = 100 feet/cms measurement = 19/20	output = 30.48 metres, 39.37 inches
Measurement = 20/30	Length = 100 cms/inches measurement = 29/30	output = 39.37 inches, 254.0 centimetres
Measurement = 30/Invalid	Length = 100 inches measurement = 39/40	output = 254.0 centimetres, “Invalid measurement”,

White Box Testing:

Submodule: convertString()
Imports: INPUT(str), upper_lower(str)
Exports: OUTPUT (str)

"Identify whether numeric values are in a given input parameter."

Category	Test Data	Expected Result
if upper_lower == 'U' or upper_lower == 'u' – selects upper case	INPUT = “Interstellar” upper_lower = “U”	OUTPUT = “ INTERSTELLAR ”
elif upper_lower == 'L' or upper_lower == 'l' – selects lower case	INPUT = “Interstellar” upper_lower = “L”	OUTPUT =“interstellar”
Enter else statement: (upper_lower != 'L' or upper_lower != 'l' and upper_lower != 'U' or upper_lower != 'u')	INPUT = “Interstellar” upper_lower = “O”	OUTPUT = ValueError("Invalid choice: must be 'U' or 'L'")

Submodule: IsValidNumber()
Imports: INPUT(str)
Exports: ValidNumber(bool)

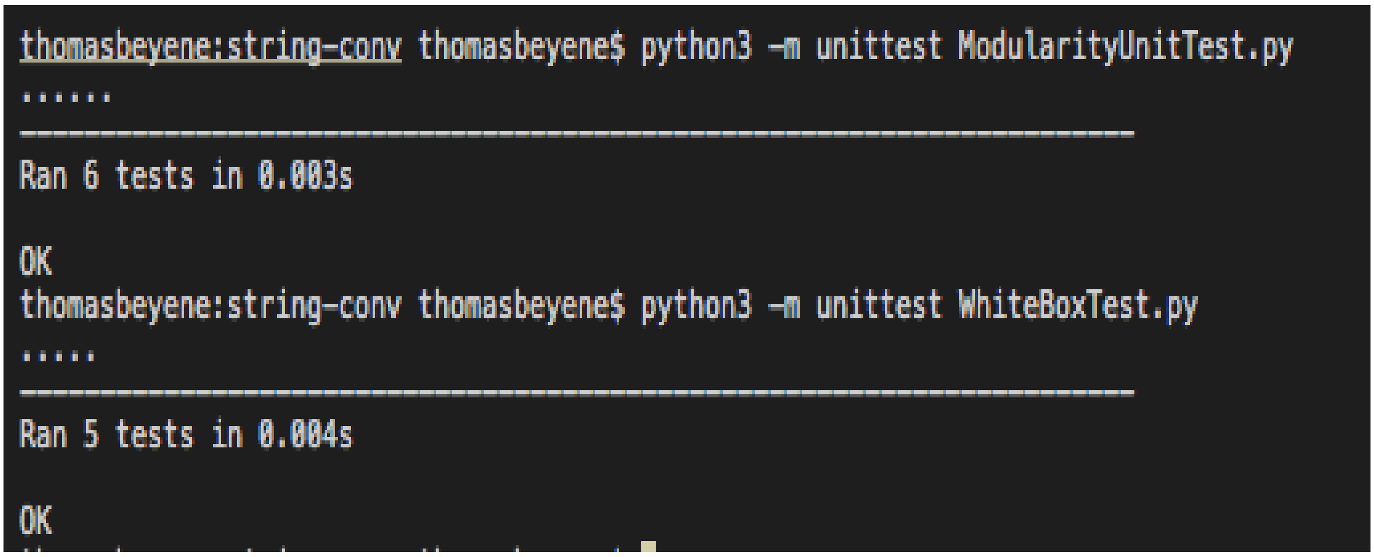
Identify whether numeric values are in a given string.

Category	Test Data	Expected Result
Enter valid String	INPUT = “3053”	OUTPUT = True
Enter invalid String	INPUT = “Interstellar”	OUTPUT = False

Test implementation and execution:

Module name	BB test design (EP)	BB test design (BVA)	WB test design	EP test code (implemented/run)	BVA test code (implemented/run)	White-Box testing (implemented/run)
convertString()	done	not done	done	done	not done	done
ContainsNumbericValue()	done	not done	not done	done	not done	not done
IsValidNumber()	done	not done	done	done	not done	done
RemoveNumeric()	done	not done	not done	done	not done	not done
RemoveNumericAndConvert()	done	not done	not done	done	not done	not done
ConvertLength()	not done	done	not done	not done	done	not done

The test cases implemented are in two different files, ModularityUnitTest.py and WhiteBoxTest.py. ModularityUnitTest.py - contains the equivalence partitioning and boundary value analysis unit test cases and the WhiteBoxTest.py contains the white box unit test cases. To test the code enter python3 -m unittest ModularityUnitTest.py and python3 -m unittest WhiteBoxTest.pypython3 -m unittest WhiteBoxTest.py. All the test conducted were successful



```
thomasbeyene:string-conv thomasbeyene$ python3 -m unittest ModularityUnitTest.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.003s

OK
thomasbeyene:string-conv thomasbeyene$ python3 -m unittest WhiteBoxTest.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.004s

OK
```

Ethics:

A scenario the code designed can be used in a large software project is implementing electronic healthcare records. There could be a lack of ethical and professional in this scenario as the software engineering team doesn't adhere to having a honest approach as they don't complete the code to the required standard and therefore didn't write comprehensive test cases as they believe the program is to the required standards. Unfortunately there was a bug that didn't remove any numeric values within given information as the team forget to complete the loop statement. The software engineering team didn't comprehensively tests the code for any bugs and were unable to locate the
bug.

This causes inaccuracy in data within the electronic health records, which were found within medical prescriptions, physical examination analysis, dosage amounts, etc. This is very dangerous as it could even risk patients lives, for example patients are prescribed medicine that they are allergic to. It increases the liability for clinicians and organisations. This has negative impacts on stakeholders, less people trust the validity of these health organisations.

This evidence reveals that developing software systems without clearly considering ethical concerns, especially in the early phase of software development life cycle phases, i.e., requirements engineering, may lead to software that could fail to fulfils societal value and may cause damage to the public at large

This should be a focus within the early stages of software development. This is done by implementing ACS Ethical guidelines when developing the software. For example developers have the duty to identify and understand the vulnerability of such hazardous requirements, making sure that they are securely operated. This is by having rigorous and comprehensive testing for the programs, to ensure public safety.

There should also be a clear, concise software development plan by the software team, where ethical concerns, are addressed in early stages of the life cycle. They should ensure there are honest, hard-working developers who are willing to learn and improve, which builds a place for shared values and reliance, not working for monetary gains. Additionally ensure there are backup plans in case something goes wrong, this ensures there isn't a long waiting time for patients as development process could become tedious. This therefore ensures public interest are in check, as waiting times are faster for patients. Not implementing ethical guidelines within the software may lead to software that could fail to fulfils societal value and may cause damage to the public.

Discussion:

Overall I was pleased with the overall completion of my assignment, I feel that I completed all tasks well, except for version control task, especially with the assignment being a lot tougher than I expected.

All the production code managed to run as expected and I managed to make all my tests pass which I am pleased about. I feel like I could have elaborated more about my code and modularity in my report and that I would work on for my future assignments.