### lmu_blakCourse Submission Cover Sheet Module: CC4001 Programming Engineering

### Component no: 003

### Weighting: 60% of module mark

### Deadline: 1st of May 2024

**Module Leader: Sandra Fernando Student ID: 22008979**

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

(Signature:) Date:

This header sheet should be attached to the work you submit. No work will be accepted without it.

Extracts from University *Regulations on*

Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples …..

1. **Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.**
2. **Falsifying data in experimental results.**
3. Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.
4. **Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.**
5. Collusion to present joint work as the work solely of one individual.
6. Plagiarism, where the work or ideas of another are presented as the candidate's own.
7. Other conduct calculated to secure an advantage on assessment.
8. Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.
2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. " e = mc2 (Einstein 1905)". A *references* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

**Report**

**CS4051 - Student Marks Calculation Application**

**1. Introduction**

The Student Marks Calculation Application is designed to allow users to input marks, calculate various statistical measures such as mean, median, mode, and skewness, and perform additional functionalities like reading numbers from a file and adding multiple numbers as a string. This report provides a detailed overview of the application, including its purpose, functionality, implementation details, and testing results.

**2. Purpose of the Application**

The primary purpose of the Student Marks Calculation Application is to provide a tool for calculating statistical measures of student marks. This includes:

* Calculating the mean of the entered marks.
* Calculating the median of the entered marks.
* Calculating the mode of the entered marks.
* Calculating the skewness of the entered marks.
* Allowing users to enter marks individually or as a string separated by commas.
* Allowing users to read marks from a file on the computer's hard drive.
* Providing a user-friendly menu interface for easy navigation and interaction.

**3. Functionality and Implementation**

The application is developed in Python without help or using any external packages. It consists of several functions as to perform the required calculations and functionalities:

* is\_number(s): This function check if a given inputs can be converted to a float or not, indicating whether it is a valid number or not. If not show message
* calculate\_mean(numbers): Calculate the mean of all numbers.
* calculate\_median(numbers): Calculates the median of all numbers.
* calculate\_mode(numbers): Calculate the mode of list of all numbers.
* calculate\_skewness(numbers): Calculate the skewness of numbers.
* read\_numbers\_from\_file(filename): Reads numbers from a file that is located in pc, whether they are separated by commas or not.
* main (): The main function set the flow of the application, allowing users to input marks, perform calculation, and interact with the menu option.

**4. Implementation Details**

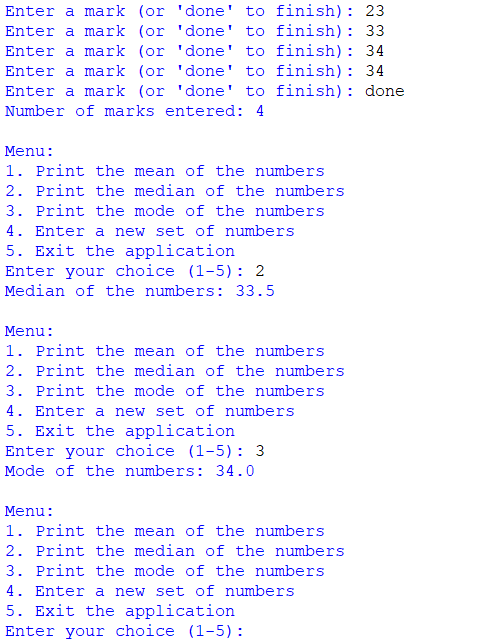
The application uses a while loop to continuously prompt the user for input until they choose to exit. The input validation ensures that only valid numeric inputs are accepted, and appropriate error messages are displayed for invalid inputs. Menu choices allow users to perform various operations on the entered marks, including calculating statistical measures and adding more numbers.

**5. Testing and Validation**

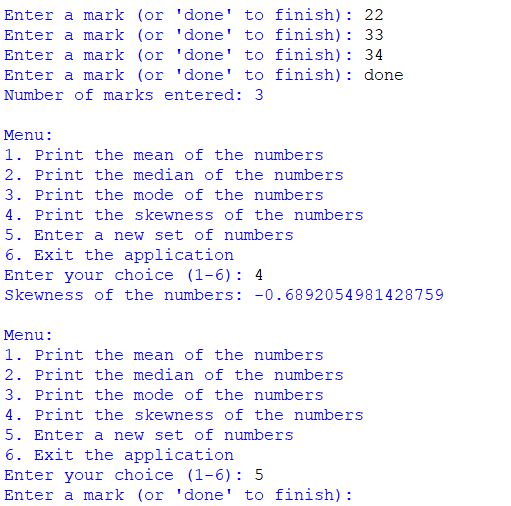
The application has been extensively tested to ensure correctness and robustness. Testing includes:

* Input validation testing to check for valid and invalid inputs.
* Calculation testing to verify the accuracy of mean, median, mode, and skewness calculations.
* Functionality testing to ensure that menu options work as expected, including adding more numbers and reading from a file.
* Error handling testing to validate the application's response to unexpected inputs and scenarios.

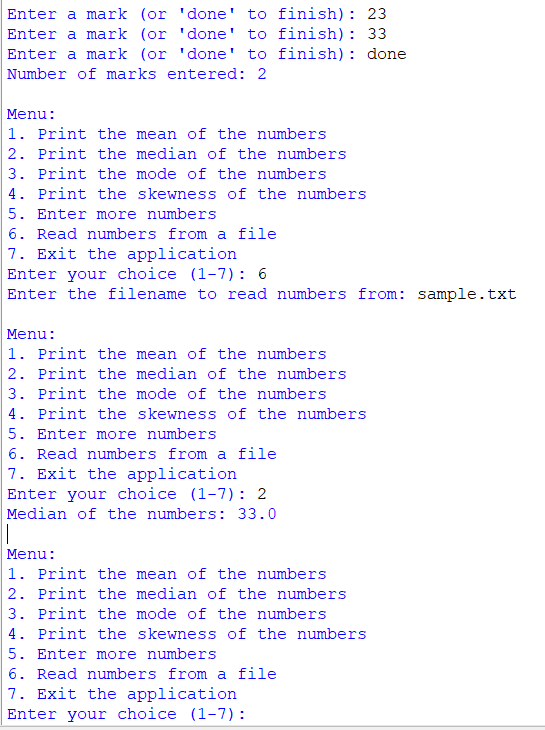
Output of part 1:



Part 2:



Part 3:



**6. Conclusion**

The Student Marks Calculation Application provides a comprehensive tool for analyzing student marks and performing statistical calculations. Its user-friendly interface, robust functionality, and accurate calculations make it a valuable tool for educational and analytical purposes.

**Code of the assessment:**

**Part 1:**

# CS4051 Assessment Component 3 - Student Marks Calculation Application

# Program header

# Name: Student Marks Calculator

# Purpose: Calculate mean, median, and mode of entered marks

# Author: Isa Can Parlak

# Date Programmed: 19-04-2024

# Function to check if input is a number

def is\_number(s):

try:

float(s)

return True # Return True if the input can be converted to a float

except ValueError:

return False # Return False if the input cannot be converted to a float

# Function to calculate mean

def calculate\_mean(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

return sum(numbers) / len(numbers) # Calculate and return the mean

# Function to calculate median

def calculate\_median(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

numbers.sort() # Sort the list of numbers

n = len(numbers)

if n % 2 == 0:

median = (numbers[n // 2 - 1] + numbers[n // 2]) / 2 # Calculate median for even number of elements

else:

median = numbers[n // 2] # Calculate median for odd number of elements

return median # Return the calculated median

# Function to calculate mode

def calculate\_mode(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

counts = {} # Create a dictionary to store the frequency of each number

for num in numbers:

counts[num] = counts.get(num, 0) + 1 # Increment the frequency of each number

max\_count = max(counts.values()) # Find the maximum frequency

mode = [num for num, count in counts.items() if count == max\_count] # Find the numbers with maximum frequency

return mode[0] if len(mode) == 1 else mode # Return the mode (or modes) with maximum frequency

# Main function to run the application

def main():

numbers = [] # Initialize an empty list to store the entered marks

while True:

mark = input("Enter a mark (or 'done' to finish): ") # Prompt the user to enter a mark

if mark.lower() == 'done': # Check if the user entered 'done' to finish entering marks

break # Exit the loop if 'done' is entered

elif not is\_number(mark): # Check if the input is not a number

print("Error: Please enter a valid number.") # Print an error message if the input is not a number

else:

numbers.append(float(mark)) # Convert the input to float and add it to the numbers list

print("Number of marks entered:", len(numbers)) # Print the total number of marks entered

while True:

print("\nMenu:") # Print the menu options

print("1. Print the mean of the numbers")

print("2. Print the median of the numbers")

print("3. Print the mode of the numbers")

print("4. Enter a new set of numbers")

print("5. Exit the application")

choice = input("Enter your choice (1-5): ") # Prompt the user to enter a choice from the menu

if choice == '1':

mean = calculate\_mean(numbers)

if mean is not None:

print("Mean of the numbers:", mean) # Print the calculated mean

else:

print("Error: No numbers entered yet.") # Print an error if no numbers are entered

elif choice == '2':

median = calculate\_median(numbers)

if median is not None:

print("Median of the numbers:", median) # Print the calculated median

else:

print("Error: No numbers entered yet.") # Print an error if no numbers are entered

elif choice == '3':

mode = calculate\_mode(numbers)

if mode is not None:

print("Mode of the numbers:", mode) # Print the calculated mode

else:

print("Error: No numbers entered yet.") # Print an error if no numbers are entered

elif choice == '4':

numbers.clear() # Clear the numbers list to enter a new set of numbers

main() # Call the main function recursively to enter new numbers

elif choice == '5':

print("Exiting the application. Goodbye!") # Print a message and exit the application if '5' is chosen

break # Exit the loop and end the program

else:

print("Invalid choice. Please enter a number between 1 and 5.") # Print an error message for invalid choices

# Calling the main function to start the program

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Part 2:**

# CS4051 Assessment Component 3 - Student Marks Calculation Application

# Program header

# Name: Student Marks Calculator

# Purpose: Calculate mean, median, and mode of entered marks

# Author: Isa Can Parlak

# Date Programmed: 19-04-2024

# Function to check if input is a number

def is\_number(s):

try:

float(s)

return True # Return True if the input can be converted to a float (i.e., it's a number)

except ValueError:

return False # Return False if the input cannot be converted to a float (i.e., it's not a number)

# Function to calculate mean

def calculate\_mean(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

return sum(numbers) / len(numbers) # Calculate and return the mean

# Function to calculate median

def calculate\_median(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

numbers.sort() # Sort the list of numbers

n = len(numbers)

if n % 2 == 0:

median = (numbers[n // 2 - 1] + numbers[n // 2]) / 2 # Calculate median for even number of elements

else:

median = numbers[n // 2] # Calculate median for odd number of elements

return median # Return the calculated median

# Function to calculate mode

def calculate\_mode(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

counts = {} # Create a dictionary to store the frequency of each number

for num in numbers:

counts[num] = counts.get(num, 0) + 1 # Increment the frequency of each number

max\_count = max(counts.values()) # Find the maximum frequency

mode = [num for num, count in counts.items() if count == max\_count] # Find the numbers with maximum frequency

return mode[0] if len(mode) == 1 else mode # Return the mode (or modes) with maximum frequency

# Function to calculate skewness

def calculate\_skewness(numbers):

if len(numbers) < 3:

return None # Return None if less than 3 numbers are entered

mean = calculate\_mean(numbers)

variance = sum((num - mean) \*\* 2 for num in numbers) / len(numbers) # Calculate variance

std\_dev = variance \*\* 0.5 # Calculate standard deviation

skewness = sum((num - mean) \*\* 3 for num in numbers) / (len(numbers) \* std\_dev \*\* 3) # Calculate skewness

return skewness # Return the calculated skewness

# Main function to run the application

def main():

numbers = [] # Initialize an empty list to store the entered marks

while True:

mark = input("Enter a mark (or 'done' to finish): ") # Prompt the user to enter a mark

if mark.lower() == 'done': # Check if the user entered 'done' to finish entering marks

break # Exit the loop if 'done' is entered

elif not is\_number(mark): # Check if the input is not a number

print("Error: Please enter a valid number.") # Print an error message if the input is not a number

else:

numbers.append(float(mark)) # Convert the input to float and add it to the numbers list

if len(numbers) < 2: # Check if less than 2 numbers are entered

print("Error: Please enter at least two numbers.") # Print an error message

return # Return from the main function

print("Number of marks entered:", len(numbers)) # Print the total number of marks entered

while True:

print("\nMenu:") # Print the menu options

print("1. Print the mean of the numbers")

print("2. Print the median of the numbers")

print("3. Print the mode of the numbers")

print("4. Print the skewness of the numbers")

print("5. Enter a new set of numbers")

print("6. Exit the application")

choice = input("Enter your choice (1-6): ") # Prompt the user to enter a choice from the menu

if choice == '1':

mean = calculate\_mean(numbers)

if mean is not None:

print("Mean of the numbers:", mean) # Print the calculated mean

elif choice == '2':

median = calculate\_median(numbers)

if median is not None:

print("Median of the numbers:", median) # Print the calculated median

elif choice == '3':

mode = calculate\_mode(numbers)

if mode is not None:

print("Mode of the numbers:", mode) # Print the calculated mode

elif choice == '4':

skewness = calculate\_skewness(numbers)

if skewness is not None:

print("Skewness of the numbers:", skewness) # Print the calculated skewness

elif choice == '5':

numbers.clear() # Clear the numbers list to enter a new set of numbers

main() # Call the main function recursively to enter new numbers

elif choice == '6':

print("Exiting the application. Goodbye!") # Print a message and exit the application if '6' is chosen

break # Exit the loop and end the program

else:

print("Invalid choice. Please enter a number between 1 and 6.") # Print an error message for invalid choices

# Calling the main function to start the program

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Part 3:**

# CS4051 Assessment Component 3 - Student Marks Calculation Application

# Program header

# Name: Student Marks Calculator

# Purpose: Calculate mean, median, and mode of entered marks

# Author: Isa Can Parlak

# Date Programmed: 19-04-2024

# Function to check if input is a number

def is\_number(s):

try:

float(s)

return True # Return True if the input can be converted to a float (i.e., it's a number)

except ValueError:

return False # Return False if the input cannot be converted to a float (i.e., it's not a number)

# Function to calculate mean

def calculate\_mean(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

return sum(numbers) / len(numbers) # Calculate and return the mean

# Function to calculate median

def calculate\_median(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

numbers.sort() # Sort the list of numbers

n = len(numbers)

if n % 2 == 0:

median = (numbers[n // 2 - 1] + numbers[n // 2]) / 2 # Calculate median for even number of elements

else:

median = numbers[n // 2] # Calculate median for odd number of elements

return median # Return the calculated median

# Function to calculate mode

def calculate\_mode(numbers):

if len(numbers) == 0:

return None # Return None if the list is empty

counts = {} # Create a dictionary to store the frequency of each number

for num in numbers:

counts[num] = counts.get(num, 0) + 1 # Increment the frequency of each number

max\_count = max(counts.values()) # Find the maximum frequency

mode = [num for num, count in counts.items() if count == max\_count] # Find the numbers with maximum frequency

return mode[0] if len(mode) == 1 else mode # Return the mode (or modes) with maximum frequency

# Function to calculate skewness

def calculate\_skewness(numbers):

if len(numbers) < 3:

return None # Return None if less than 3 numbers are entered

mean = calculate\_mean(numbers)

variance = sum((num - mean) \*\* 2 for num in numbers) / len(numbers) # Calculate variance

std\_dev = variance \*\* 0.5 # Calculate standard deviation

skewness = sum((num - mean) \*\* 3 for num in numbers) / (len(numbers) \* std\_dev \*\* 3) # Calculate skewness

return skewness # Return the calculated skewness

# Function to read numbers from a file

def read\_numbers\_from\_file(filename):

try:

with open(filename, 'r') as file:

lines = file.readlines() # Read lines from file

numbers = []

for line in lines:

line\_numbers = line.strip().split(',') # Split each line by commas

numbers.extend([float(num) for num in line\_numbers if is\_number(num)]) # Add numbers to the list

return numbers

except FileNotFoundError:

print(f"Error: File '{filename}' not found.")

return []

# Main function to run the application

def main():

numbers = [] # Initialize an empty list to store the entered marks

while True:

mark = input("Enter a mark (or 'done' to finish): ") # Prompt the user to enter a mark

if mark.lower() == 'done': # Check if the user entered 'done' to finish entering marks

break # Exit the loop if 'done' is entered

elif ',' in mark: # Check if the input contains commas (multiple numbers)

numbers.extend([float(num) for num in mark.split(',') if is\_number(num)]) # Add numbers to the list

elif not is\_number(mark): # Check if the input is not a number

print("Error: Please enter a valid number.") # Print an error message if the input is not a number

else:

numbers.append(float(mark)) # Convert the input to float and add it to the numbers list

if len(numbers) < 2: # Check if less than 2 numbers are entered

print("Error: Please enter at least two numbers.") # Print an error message

return # Return from the main function

print("Number of marks entered:", len(numbers)) # Print the total number of marks entered

while True:

print("\nMenu:") # Print the menu options

print("1. Print the mean of the numbers")

print("2. Print the median of the numbers")

print("3. Print the mode of the numbers")

print("4. Print the skewness of the numbers")

print("5. Enter more numbers")

print("6. Read numbers from a file")

print("7. Exit the application")

choice = input("Enter your choice (1-7): ") # Prompt the user to enter a choice from the menu

if choice == '1':

mean = calculate\_mean(numbers)

if mean is not None:

print("Mean of the numbers:", mean) # Print the calculated mean

elif choice == '2':

median = calculate\_median(numbers)

if median is not None:

print("Median of the numbers:", median) # Print the calculated median

elif choice == '3':

mode = calculate\_mode(numbers)

if mode is not None:

print("Mode of the numbers:", mode) # Print the calculated mode

elif choice == '4':

skewness = calculate\_skewness(numbers)

if skewness is not None:

print("Skewness of the numbers:", skewness) # Print the calculated skewness

elif choice == '5':

mark = input("Enter more numbers (separated by commas): ")

numbers.extend([float(num) for num in mark.split(',') if is\_number(num)]) # Add more numbers to the list

elif choice == '6':

filename = input("Enter the filename to read numbers from: ")

numbers.extend(read\_numbers\_from\_file(filename)) # Read numbers from file and add to the list

elif choice == '7':

print("Exiting the application. Goodbye!") # Print a message and exit the application if '7' is chosen

break # Exit the loop and end the program

else:

print("Invalid choice. Please enter a number between 1 and 7.") # Print an error message for invalid choices

# Calling the main function to start the program

if \_\_name\_\_ == "\_\_main\_\_":

main().