Software Design and Development

Assessment Report

**Task 1 Evidence:**

**Algorithm**

**#1.start**

**#2.initialize coursework\_mark=0 and prelim\_mark=0 to validate the input values**

**#3.while not coursework\_mark is integer and 0 <= coursework\_mark <= 60**

**#4 display 'Enter coursework mark'**

**#5 coursework\_mark = input()**

**#6.if not coursework\_mark is integer and 0 <= coursework\_mark <= 60**

**#7.display 'Invalid input'**

**#8.end while loop**

**#9. while not prelim\_\_mark is integer and 0 <= prelim\_mark <= 60**

**#10 display 'Enter prelim mark'**

**#11 prelim\_mark = input()**

**#12 if not prelim\_mark is integer and 0 <= prelim\_mark <= 60**

**#13 display 'Invalid input'**

**#14 end while loop**

**#15.percentage =((coursework\_mark+prelim\_mark)\*100)/150**

**#16 if percentage >=70**

**#17.display 'A Grade'**

**#18.else if percentage >= 60 and percentage < 70**

**#19. display 'B Grade'**

**#20.else if percentage >= 50 and percentage < 60**

**#21. display 'C Grade'**

**#22.else if percentage >= 45 and percentage < 50**

**#23. display 'D Grade'**

**#24.else**

**#25. display 'No Grade'**

**#26.end**

**Task 2 Evidence:**

**Using the table below complete the test plan for the program:**

**Note: You may need to add additional rows in order to show a complete set of tests.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Reason for test (valid/invalid –Normal,exceptional,extreme etc.)** | **Test Data** | **Expected Output** |
| Check student name | Valid | Get student name | Student name |
| Check coursework mark | Valid | Get coursework mark | Coursework mark |
| Check prelim mark | Valid | Get prelim mark | Prelim mark |
| If Grade >=70 | Normal | If percentage is 70% or over | Student name and A grade |
| Elif Grade>=60 and 69 | Normal | If percentage is 60% and over | Student name and B grade |
| Elif Grade>=50 and 59 | Normal | If percentage is 50% and over | Student name and C Grade |
| Elif Grade>=45 and 49 | Valid | If percentage is 45% and 49% | Student name and D grade |
| Else | Valid | If percentage is 45% and 49% | Student name and no grade |

**Task 3 and 4 Evidence:**

**Copy/ paste your source code and add below: Make sure to add internal commentary asked for.**

#Author Pranoti

#Date 06/03/2025

**#Calculate Student Percentage using Functions**

def validateinputs(course\_mark,prelim\_mark):

    if prelim\_mark.isdigit() or course\_mark.isdigit():

      return True

    else:

      print("Invalid marks.Please give marks in positive integer number")

      return False

**#creating function to calculate Grade**

def student\_grade():

**#checking if the user entering valid info**

  if(course\_mark>60 and prelim\_mark>90): print("Invalid input..Please enter correct marks")

  else:

**#allocating grades based on the percentage**

   if (grade\_percentage >= 70 and prelim\_mark <= 100): print ("Hi",stud\_name,"You got grade A")

   elif (grade\_percentage >= 60 and grade\_percentage <= 70): print ("Hi",stud\_name,"You got grade B")

   elif (grade\_percentage >= 50 and grade\_percentage <= 60): print ("Hi",stud\_name,"You got grade C")

   elif (grade\_percentage >= 45 and grade\_percentage <=49): print ("Hi",stud\_name,"You got grade D")

   elif  (grade\_percentage <45): print("Hi",stud\_name,"You are failed and No grade")

**#while code to run the program on user choice**

**#main code starts from this line**

choice = "Y"

while(choice == "Y"):

**#Getting inputs from user**

 stud\_name=str(input("Please enter your name and marks:"))

 course\_mark = input("Enter your course marks out of 60: ")

 prelim\_mark = input("Enter your prelim marks out of 90: ")

 isValid=validateinputs(course\_mark,prelim\_mark)

**#calling the function to calculate grades**

 if(isValid):

   course\_mark = float(course\_mark)

   prelim\_mark = float(prelim\_mark)

   grade\_percentage = ((course\_mark + prelim\_mark)\*100) /150

   student\_grade()

**#calculating the percentage**

   choice=str(input("Do you want to run the program again Y/N:"))

#print message if user chooses to exit

print("You choose to exit,Good bye..!")

**A black screen with white text

AI-generated content may be incorrect.**

**Task 5 Evidence:**

**Use the table below to complete the testing outlined in above test plan.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | **Reason for test** | **Test Data** | **Expected Output** | **Actual  Output** | **Comments** |
| Check student name | Valid | Get student name | Student name | Student name | See Test 1 |
| Check coursework mark | Valid | Get coursework mark | Coursework mark | Asking for coursework mark | See Test 2 |
| Check prelim mark | Valid | Get prelim mark | Prelim mark | Asking for Prelim mark | See Test 3 |
| If Grade >=70 | Normal | If percentage is 70% or over | Student name and A grade | Student name: A Grade | See Test 4 |
| Elif Grade>=60 and 69 | Valid | If percentage is 60% and over | Student name and B grade | Student name: B Grade | See Test 5 |
| Elif Grade>=50 and 59 | Valid | If percentage is 50% and over | Student name and C Grade | Student name: C Grade | See Test 6 |
| Elif Grade>=45 and 49 | Valid | If percentage is 45% and 49% | Student name and D grade | Student name: D Grade | See Test 7 |
| Else | Valid | If percentage is 45% and 49% | Student name and no grade | Student name: NO Grade | See Test 8 |

**Test 1**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Test 2**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Test3**

**A computer screen shot of text

AI-generated content may be incorrect.**

**Test4**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Test5**

**A computer screen shot of text

AI-generated content may be incorrect.**

**Test6**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**Test7**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Task 6 Evidence:**

#Author Pranoti

#Date 06/03/2025

#Calculate Student Percentage using Files

def read\_data\_from\_file(file\_path):

    """

    Function to read students' data from an external file.

    Each line in the file should contain:

    Name, Coursework Mark (out of 60), Prelim Mark (out of 90)

    Example line: John Doe, 45, 80

    Returns a list of student records as tuples.

    """

    students = []

    with open(file\_path, 'r') as file:  # Open the file in read mode

        for line in file:  # Read each line from the file

            data = line.strip().split(",")  # Split line into components (Name, Coursework, Prelim)

            name = data[0].strip()

            coursework = float(data[1].strip())

            prelim = float(data[2].strip())

            students.append((name, coursework, prelim))  # Append student data as a tuple

    return students

def calculate\_percentage(coursework, prelim):

    """Function to calculate percentage."""

    total = coursework + prelim

    percentage = (total \* 100) / 150

    return percentage

def determine\_grade(percentage):

    """Function to determine the grade based on percentage."""

    if percentage >= 70:

        return "A"

    elif percentage >= 60:

        return "B"

    elif percentage >= 50:

        return "C"

    elif percentage >= 45:

        return "D"

    else:

        return "No Grade"

def count\_a\_grades(students):

    """

    Function to count the occurrences of "A" grades in the class.

    This implements the "Count Occurrences" standard algorithm:

    - Initialize a counter to zero.

    - Loop through the list of students.

    - If the grade is "A", increment the counter by 1.

    - Return the final count.

    """

    a\_count = 0  # Initialize counter

    for student in students:

        name, coursework, prelim = student

        percentage = calculate\_percentage(coursework, prelim)  # Calculate percentage

        grade = determine\_grade(percentage)  # Determine grade

        if grade == "A":  # Check if grade is "A"

            a\_count += 1  # Increment counter

    return a\_count  # Return the final count

def process\_students(file\_path):

    """

    Function to process student data.

    Reads data from the file, calculates percentage and grade for each student,

    counts the number of "A" grades, and displays the results.

    """

    students = read\_data\_from\_file(file\_path)  # Retrieve students' data from the file

    if not students:  # If no data was read, stop execution

        print("No student data to process.")

        return

    print(f"{'Name':<20}{'Percentage':<15}{'Grade':<10}")  # Print header

    print("-" \* 45)

    for student in students:

        name, coursework, prelim = student

        percentage = calculate\_percentage(coursework, prelim)  # Calculate percentage

        grade = determine\_grade(percentage)  # Determine grade

        print(f"{name:<20}{percentage:<15.2f}{grade:<10}")  # Display result for each student

    # Count and display the number of "A" grades

    a\_grade\_count = count\_a\_grades(students)

    print(f"\nNumber of 'A' grades in the class: {a\_grade\_count}")

def main():

    """Main function to run the program."""

    file\_path = input("Enter the path to the student data file: ")  # Get file path from the user

    process\_students(file\_path)  # Process the data for all students

# Run the program

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

    main()

**Task 7 Evidence:**

#Author Pranoti

#Date 06/03/2025

#Calculate Student Percentage using Files

#Task 7:

#Alter your program to enable it to find out how many “A” passes are in the class by using the “Count Occurrences” standard algorithm.

def readnames():

 testFile=open("namesS.txt", "r") #open file

 for line in testFile:

   stripped\_line = line.strip()

   #read a line as string and remove any spaces

   names.append(stripped\_line)

 testFile.close()

def readmark1():

 testFile=open("Mark1.txt", "r") #open file

 for line in testFile:

  stripped\_line = line.strip()

  mark1.append(stripped\_line)

 testFile.close()

def readmark2():

   testFile=open("Mark2.txt", "r") #open file

   for line in testFile:

     stripped\_line = line.strip()

     mark2.append(stripped\_line)

   testFile.close()

def CalcGrade():

  count=int(0)

  max=int(0) #we are looking for the max percentage

  print("\n");

  for x in range (len(names)):

   Grade=((float(mark1[x])+float(mark2[x]))\*100)/150

   #print(names[x], "you have got ", Grade)

   if Grade >= 70 : # if percentage is 70% or over

    print(names[x], "you have got ", Grade ,": A Grade")

   elif Grade >= 60 and 69 : # if percentage is 60% or over

    print(names[x], "you have got ", Grade,": B Grade")

   elif Grade >=50 and 59 : # if percentage is 50% or over

    print(names[x], "you have got ", Grade ,": C Grade")

   elif Grade >=45 and 49 : #if percentage is 45% and 49%

    print(names[x], "you have got ", Grade ,": D Grade")

   else: # if no match we use this option

    print(names[x], "you have got ", Grade ,": No grade")

   if Grade>=70 : # check if grade is A

    count=count+1 # count get 1 added

   if ((int(Grade) > max)):

    max=int (Grade)

    print ("The number of A passes in class are ",count)

    print ("The best percentage in the class is ", max)

#create a list

names=[]

mark1=[]

mark2=[]

Grade=[]

#maincode

readnames()

readmark1()

readmark2()

CalcGrade()

**Task 8 Evidence**

**Write your description of the fetch execute cycle below: You may include a diagram as well.**

**Answer:**

**The fetch-execute cycle is the fundamental process through which a computer executes programs. It**

**involves several components, including the processor (CPU), memory and buses.**

**Software Design and Development J27C 76**

**• Processor (CPU): It is the brain of the computer and executes instructions. It consists of Arithmetic**

**Logic Unit (ALU), Control Unit ( CU) and registers.**

**• Memory: it stores both data and instructions that the CPU needs to execute. Memory is divided into**

**RAM (Random Access Memory) and ROM (Read Only Memory)**

**• Buses: are pathways that allow data and instructions to travel between the CPU, memory and other**

**components of the computer. The primary buses involved in the fetch execute cycle are the address**

**bus, data bus and control bus.**

**The fetch-execute cycle process:**

**• Fetch:**

**The CPU fetches the next instruction from memory. It sends the memory address of the next**

**instruction over the address bus.**

**• Decode:**

**The Control Unit decodes the fetched instruction, determining what operation needs to be**

**performed and what data, if any, is required.**

**• Execute:**

**The decoded instruction is executed by the ALU or other relevant components of the CPU. This**

**could involve performing arithmetic or logical operations, accessing data from memory, or**

**transferring data between registers.**

**• Write back:**

**If the executed instruction produces a result that needs to be stored back in memory or a register,**

**the CPU writes this result back to the appropriate location.**

**This cycle continues repeatedly, with the CPU fetching, decoding, and executing instructions until the program**

**terminates or encounters an error. The coordination of these activities is facilitated by the control signals sent**

**over the control bus, ensuring that the instructions are executed in the correct sequence and at the correct time.**

**Diagram:**

**A diagram of a process

AI-generated content may be incorrect.**