Examiners' Report
Lead Examiner Feedback

January 2021

Pearson BTEC Nationals
In Computing (31771H)
Unit 4: Software Design and Development Project

January 2021
31771H_2021_ER

Introduction

This was the fifth examination series for Level 3 BTEC Computing Unit 4: Software Design and Development Project.

This unit is a paper-based exam, assessed through a task-based assessment. The set task assesses learners' ability to design, create and evaluate software using Python (3.4 or a later version) or one of the C family programming languages. This unit is a mandatory unit for all learners studying the extended diploma.

The examination for this unit will always contain five activities and each one will be linked to a scenario. The scenario is clearly stated at the beginning of each assessment. The activities will test learners on different areas of the specification, and learners are expected to apply their knowledge to the scenario.

All Activities of the examination paper provide differentiation at all attainment levels and the brief is designed to escalate in difficulty so that a larger percentage of higher-grade marks depends on the skills, knowledge, understanding and application of theory.

Introduction to the Overall Performance of the Unit

The overall performance of learners was similar to the previous series for this unit.

The performance on Activity 1 resulted in most learners picking up marks in band 2. Most of the responses used BCS symbols, the better responses were able to break down the requirements into relevant parts. Learners provided evidence of links between component parts but evidence of handling errors within the flowcharts was not always present. Many learners tried to fit the entire flowchart on to one page, which tended to make them difficult to follow.

Activity 2 was of generally of a good standard and demonstrated the learner ability to apply pseudocode design methodologies to the scenario. Some learners produced pseudocode that was very similar to the code produced in activity 4. This reduces clarity and readability for anyone not familiar with the language specific functions or syntax. The aim of the pseudocode is to provide a step in the design process that would allow a third party if needed to continue with the coding and any of the specified languages.

Activity 3 & 4 (testing) was not well demonstrated which resulted in many learners only accessing band 1. It is recommended that centres reinforce what a test plan consists of and the importance of testing throughout the whole design and development process. Some test plans seen did not include any test data which meant no marks could be awarded. In most cases, the testing carried out did not evidence any errors encountered which is essential for accessing higher marks.

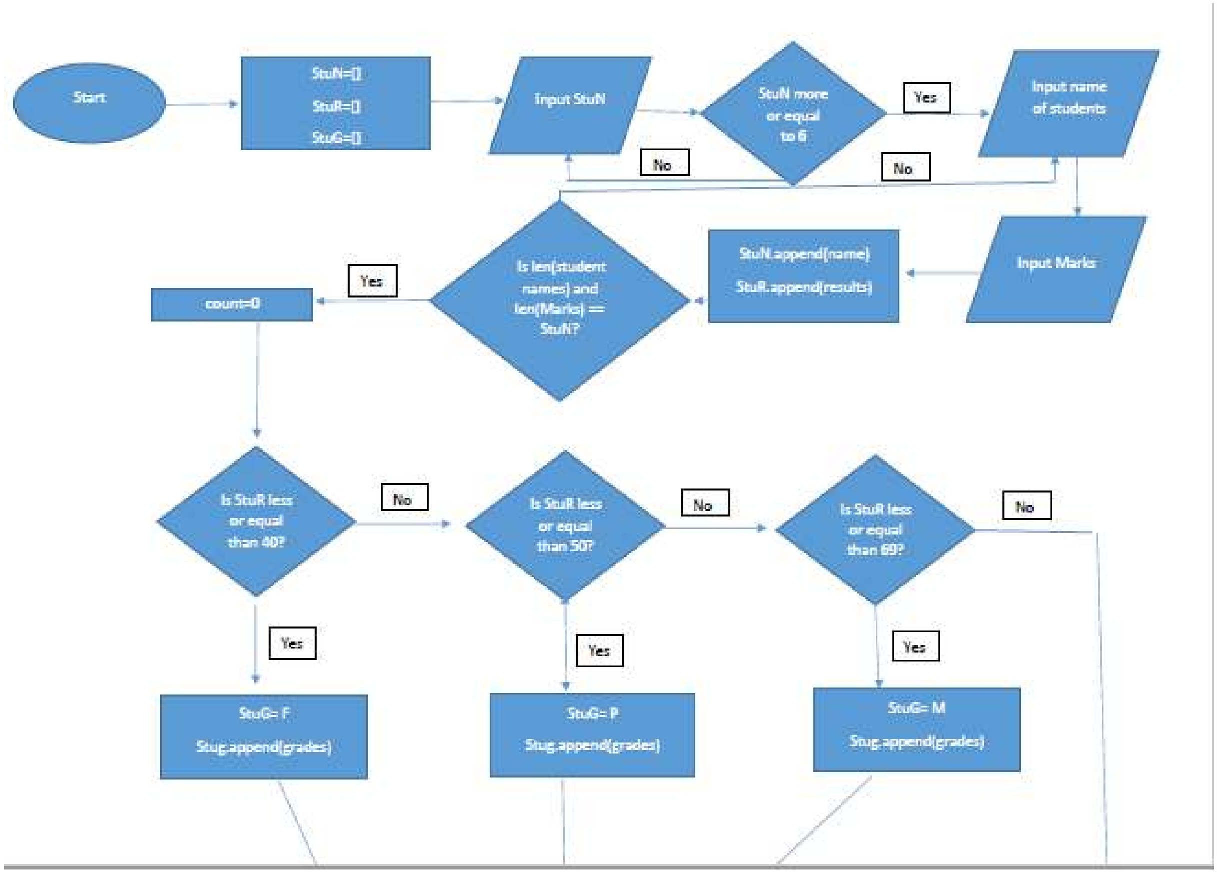
Activity 4 (Coding) was well completed by some learners with marks awarded in the top mark band as they produced a working solution along with detailed comments. These learners validated the inputs well and in trapped errors due to wrong data types being input by the user. The allocation of grades to marks was done well by some learners and the code accurately assigned the grades. Some learners found the writing of the data to a file a challenge, this stage, therefore, was not completed by a significant number of learners.

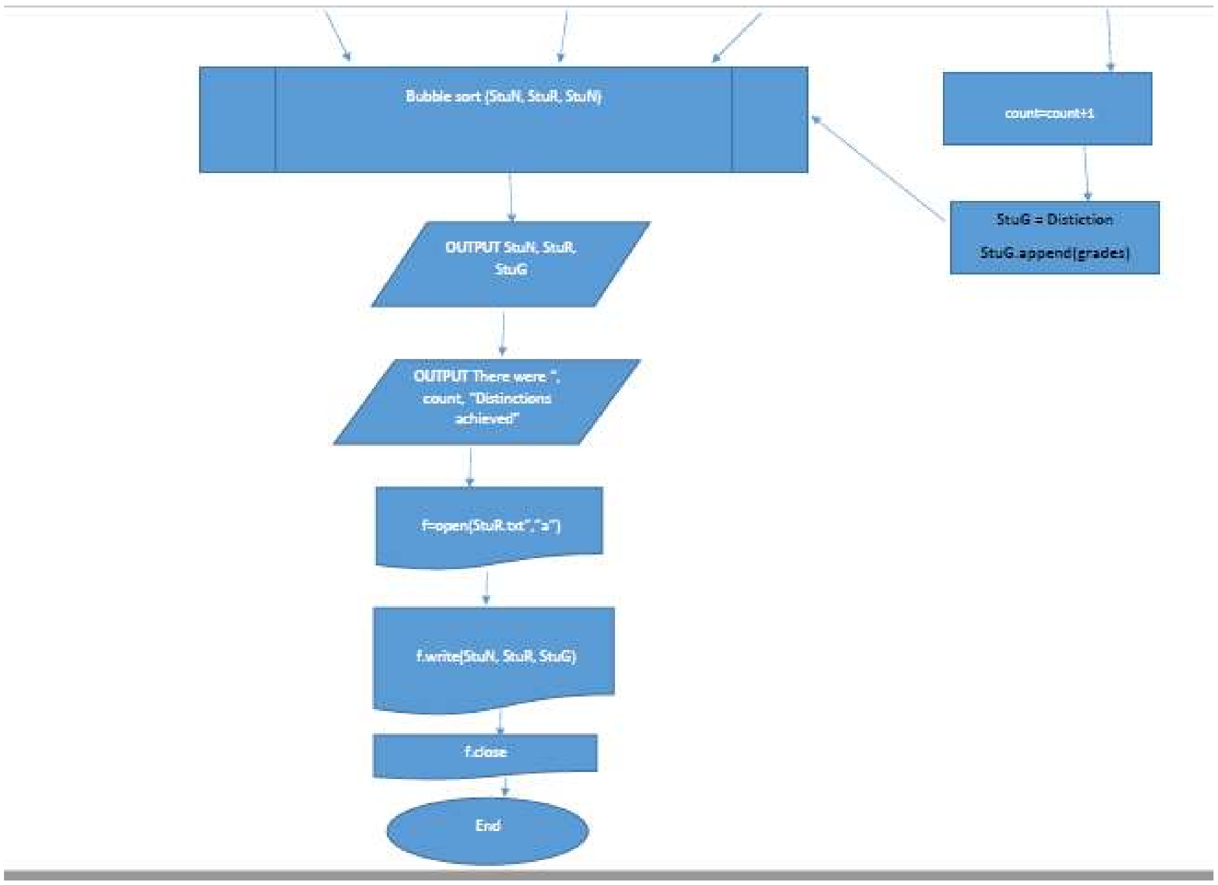
The evaluations (activity 5) were of a good standard and most learner's accessed bands two and three. Some learners only produced a step-by-step account of what they did which resulted in marks from band 1 being awarded.

Individual Questions

The following section considers each question on the paper, providing examples of learner responses and a brief commentary of why the responses gained the marks they did. This section should be considered with the live external assessment and the corresponding mark scheme.

Activity 1

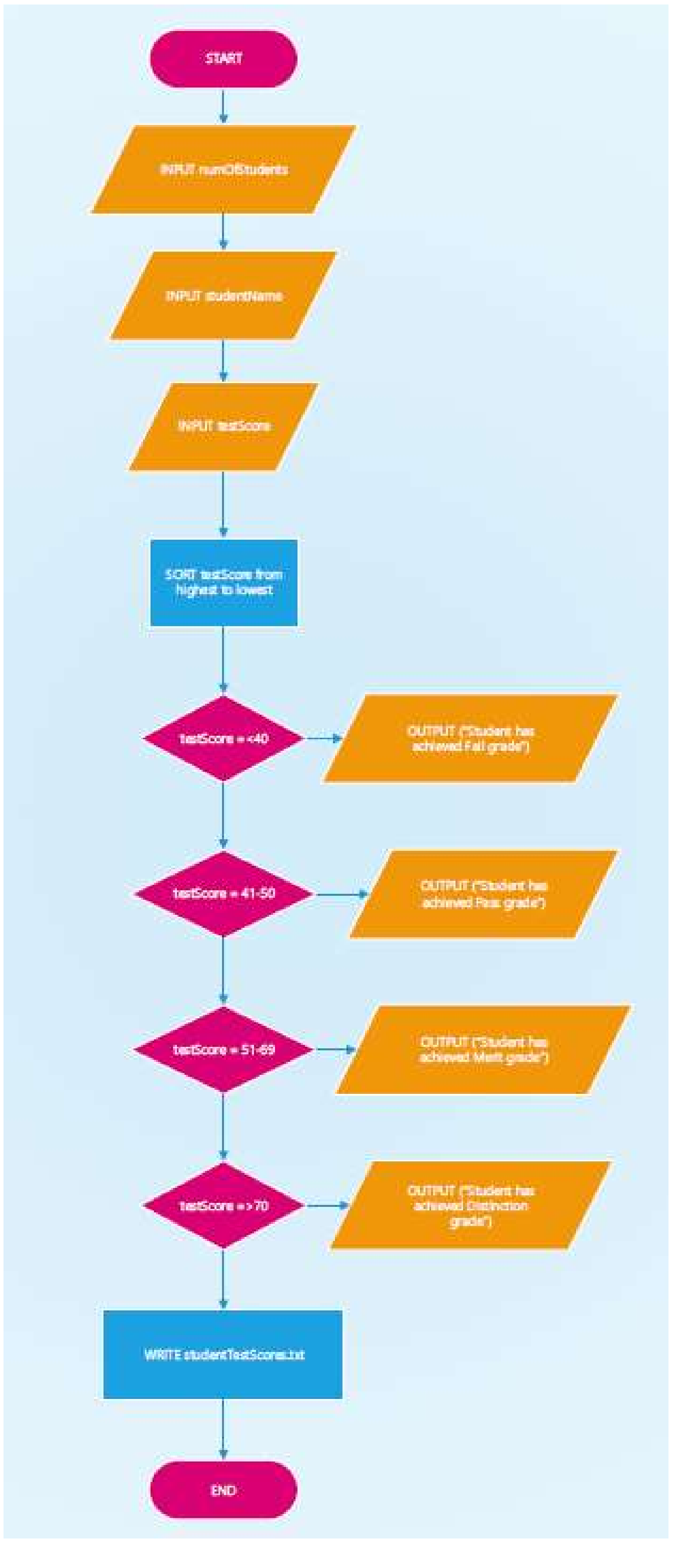



Bubble sort (StuN, StuR, StuN)
count=count+1
StuG = Distiction
StuG.append(grades)
OUTPUT StuN, StuR, StuG
OUTPUT There were ", count, "Distinctions achieved"
f=open(StuR.txt","a")
f.write(StuN, StuR, StuG)
f.close
End

Lead Examiner Commentary:

The learner shows accurate use of BCS symbols. The logic is correct, the flowchart is easy to follow and breaks down requirements into component parts. Shows full coverage of input and outputs, the naming conventions used are appropriate and consistent.
The error handling criteria are not quite complete as the procedures for handling unexpected errors are not fully demonstrated.

Band 3 (9 marks).

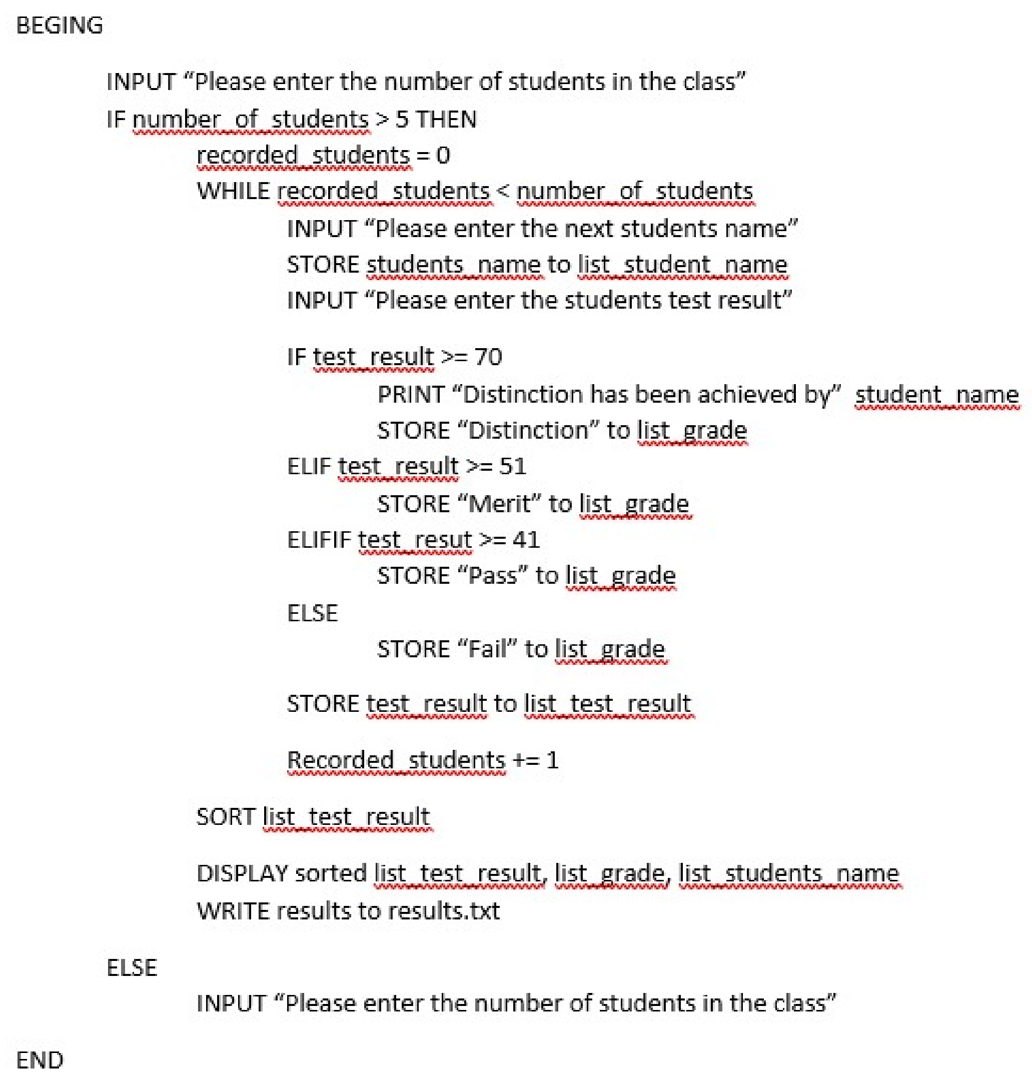

START
INPUT numOfStudents
INPUT studentName
INPUT testScore
SORT testScore from highest to lowest
testScore = <40
OUTPUT ("Student has achieved Fail grade")
testScore = 41-50
OUTPUT ("Student has achieved Pass grade")
testScore = 51-69
OUTPUT ("Student has achieved Merit grade")
testScore = >70
OUTPUT ("Student has achieved Distinction grade")
WRITE studentTestScores.txt
END

Lead Examiner Commentary:

The learner shows limited use of BCS symbols, limited coverage of inputs, the leaner is inconsistent with naming conventions and does use of Y/N on decisions in flowchart therefore not possible to follow. The routes terminate in dead ends and there is no loop structure for more than one mark to be input.

Mark in **band 1 (2 marks)**.

Activity 2

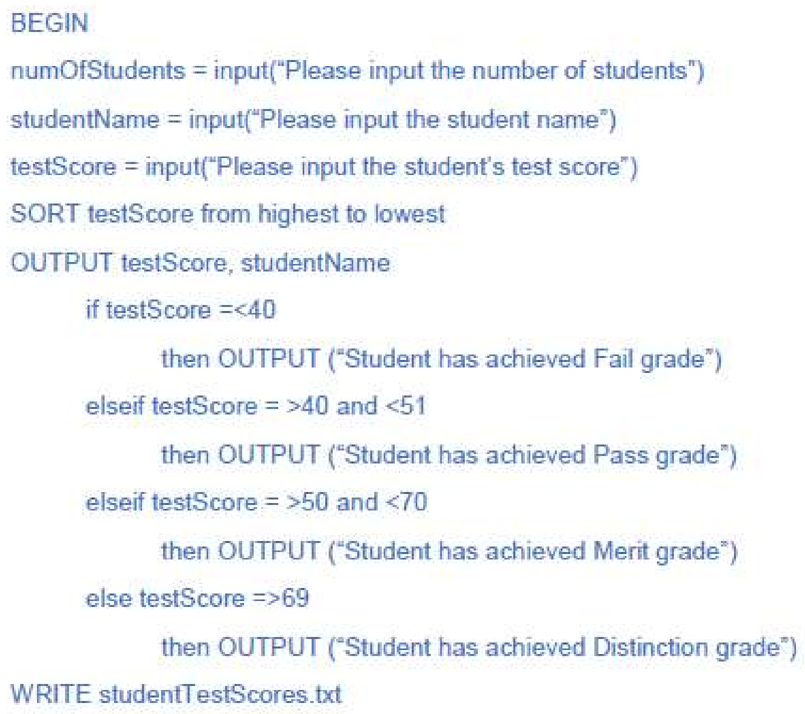


```
BEGING

    INPUT “Please enter the number of students in the class”
    IF number_of_students > 5 THEN
        recorded_students = 0
        WHILE recorded_students < number_of_students
            INPUT “Please enter the next students name”
            STORE students_name to list_student_name
            INPUT “Please enter the students test result”

            IF test_result >= 70
                PRINT “Distinction has been achieved by”  student_name
                STORE “Distinction” to list_grade
            ELIF test_result >= 51
                STORE “Merit” to list_grade
            ELIFIF test_resut >= 41
                STORE “Pass” to list_grade
            ELSE
                STORE “Fail” to list_grade

            STORE test_result to list_test_result

            Recorded_students += 1

        SORT list_test_result

        DISPLAY sorted list_test_result, list_grade, list_students_name
        WRITE results to results.txt

    ELSE
        INPUT “Please enter the number of students in the class”

END
```

Lead Examiner Commentary:

The learner has produced a structure which shows appropriate and consistent use of hierarchy and indentation, providing clarity and mostly readable pseudocode. The pseudocode will provide a working solution with some minor errors. Appropriate naming conventions have been used and precise use of logical operations.

Mark in **band 3 (9 marks)**.



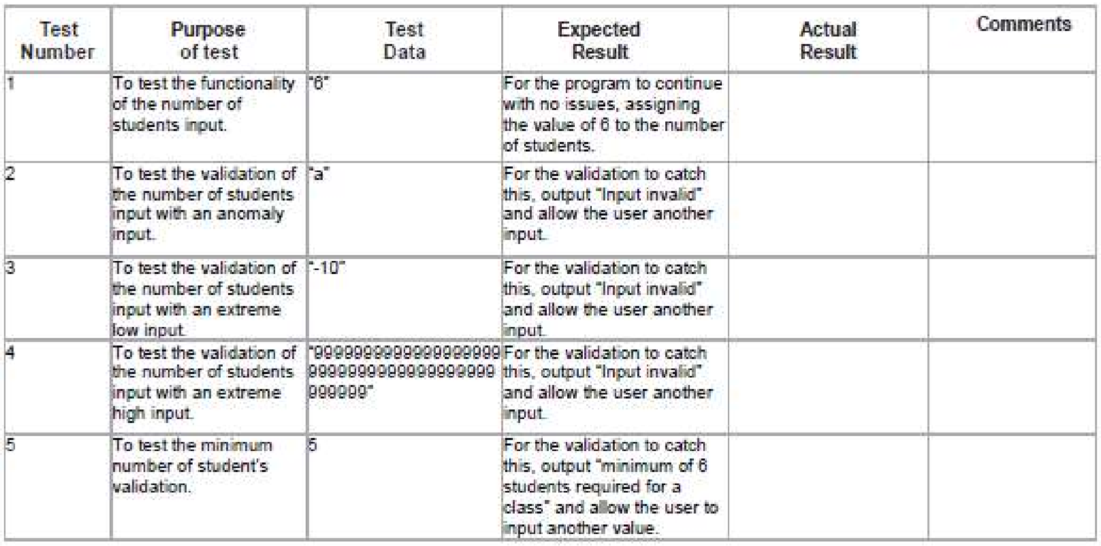
```
BEGIN
numOfStudents = input("Please input the number of students")
studentName = input("Please input the student name")
testScore = input("Please input the student's test score")
SORT testScore from highest to lowest
OUTPUT testScore, studentName
    if testScore =<40
        then OUTPUT ("Student has achieved Fail grade")
    elseif testScore = >40 and <51
        then OUTPUT ("Student has achieved Pass grade")
    elseif testScore = >50 and <70
        then OUTPUT ("Student has achieved Merit grade")
    else testScore =>69
        then OUTPUT ("Student has achieved Distinction grade")
WRITE studentTestScores.txt
```

Lead Examiner Commentary:

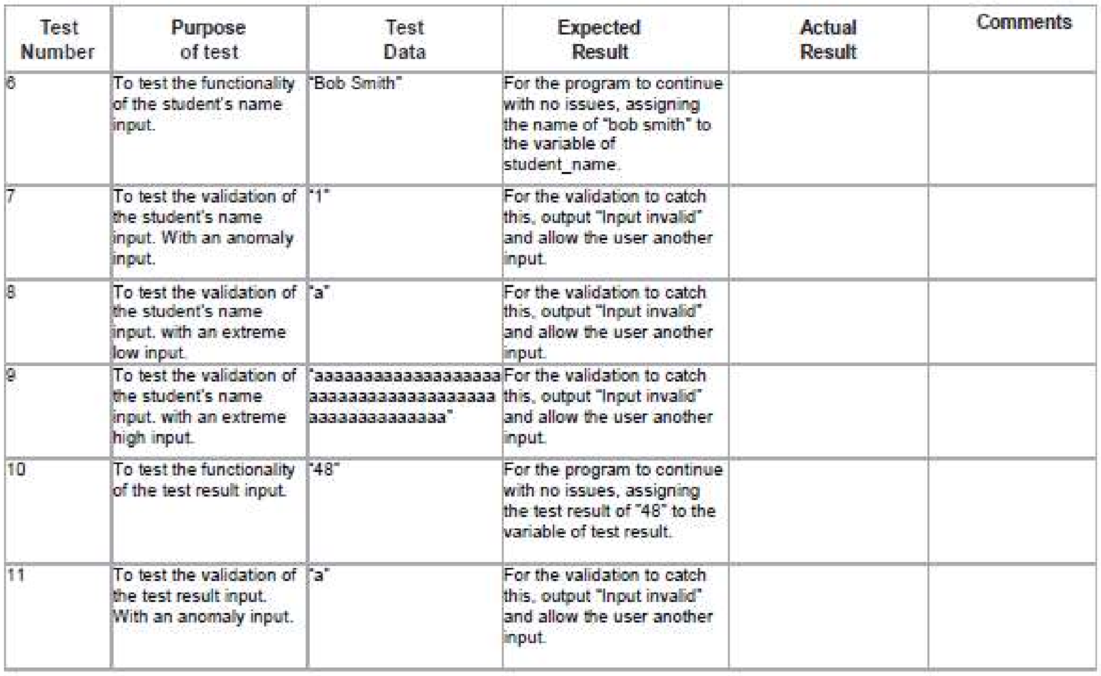
The learner has produced a structure that shows some appropriate hierarchies, readability is limited, and sequences are incomplete. The code uses appropriate naming conventions but there are inconsistencies. There is imprecise use of logical operations, leading to an incomplete solution.

Mark in **band 1 (3 marks)**.

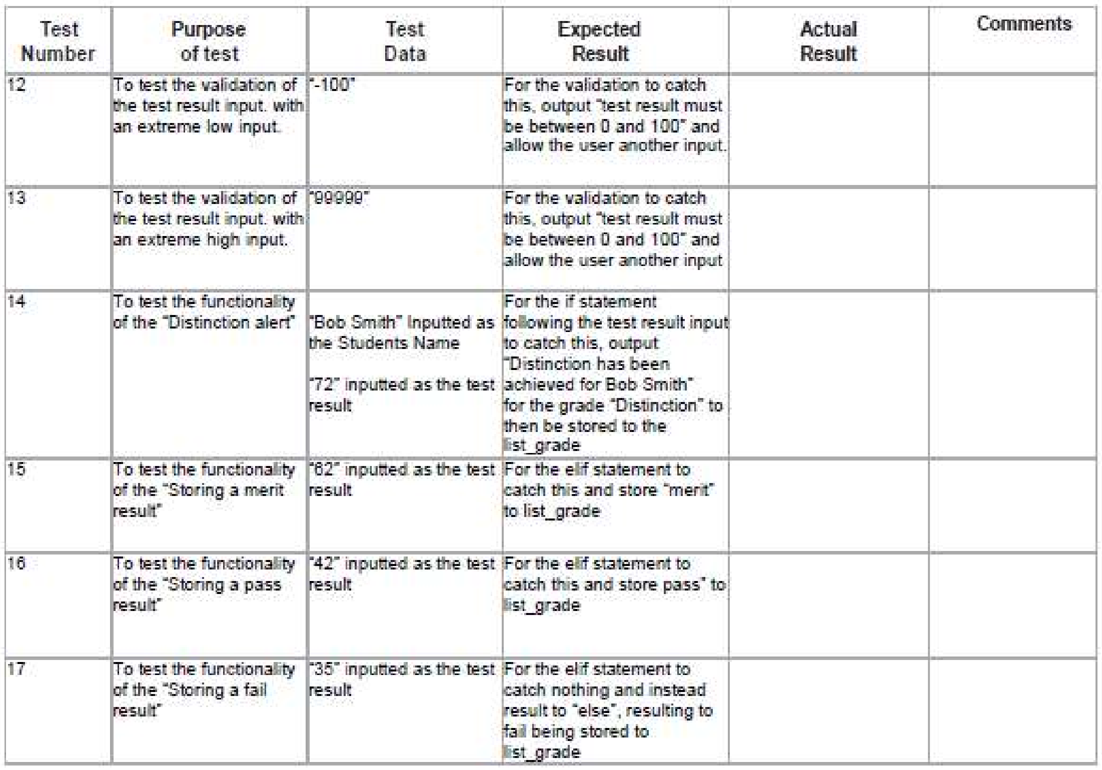
Activity 3



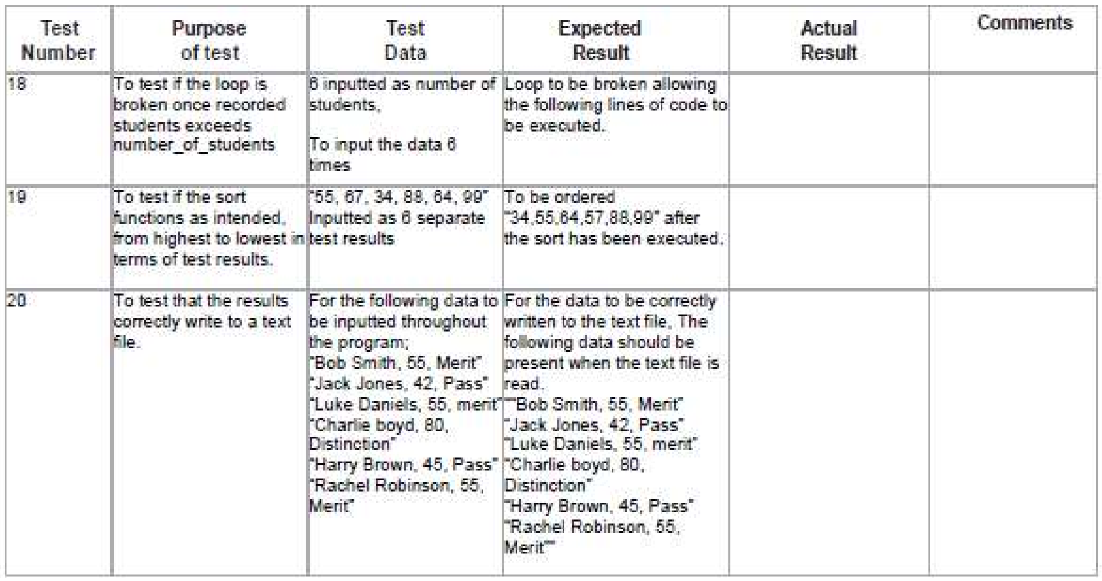
Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
1	To test the functionality of the number of students input.	"6"	For the program to continue with no issues, assigning the value of 6 to the number of students.		
2	To test the validation of the number of students input with an anomaly input.	"a"	For the validation to catch this, output "Input invalid" and allow the user another input.		
3	To test the validation of the number of students input with an extreme low input.	"-10"	For the validation to catch this, output "Input invalid" and allow the user another input.		
4	To test the validation of the number of students input with an extreme high input.	"9999999999999999999 9999999999999999999 999999"	For the validation to catch this, output "Input invalid" and allow the user another input.		
5	To test the minimum number of student's validation.	5	For the validation to catch this, output "minimum of 6 students required for a class" and allow the user to input another value.		



Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
6	To test the functionality of the student's name input.	"Bob Smith"	For the program to continue with no issues, assigning the name of "bob smith" to the variable of student_name.		
7	To test the validation of the student's name input. With an anomaly input.	"1"	For the validation to catch this, output "Input invalid" and allow the user another input.		
8	To test the validation of the student's name input. with an extreme low input.	"a"	For the validation to catch this, output "Input invalid" and allow the user another input.		
9	To test the validation of the student's name input. with an extreme high input.	"aaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaa"	For the validation to catch this, output "Input invalid" and allow the user another input.		
10	To test the functionality of the test result input.	"48"	For the program to continue with no issues, assigning the test result of "48" to the variable of test result.		
11	To test the validation of the test result input. With an anomaly input.	"a"	For the validation to catch this, output "Input invalid" and allow the user another input.		



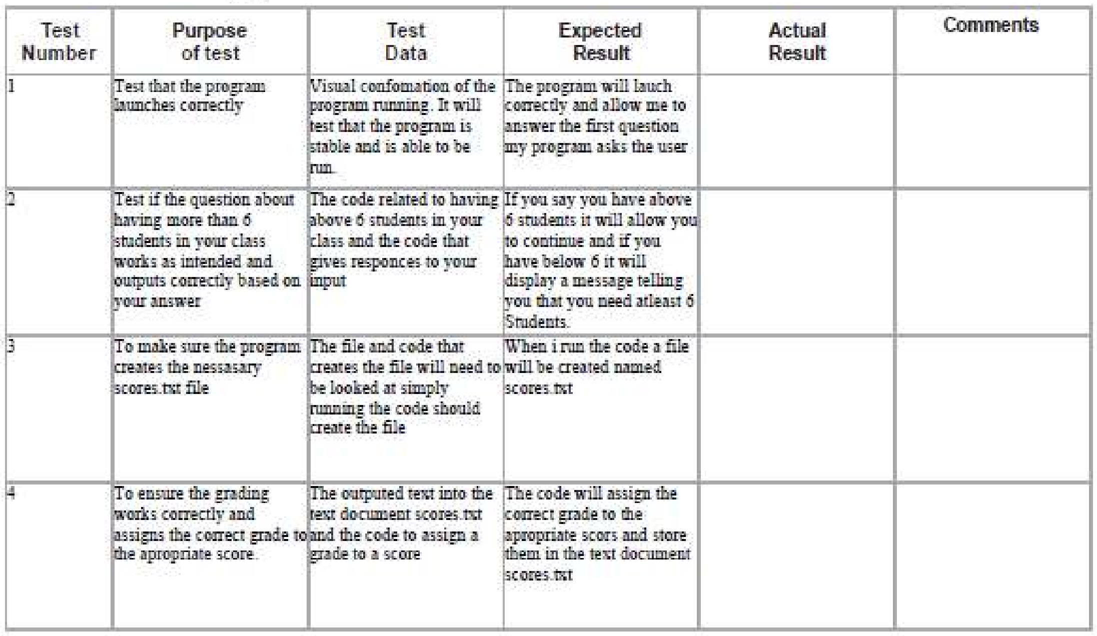
Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
12	To test the validation of the test result input. with an extreme low input.	"-100"	For the validation to catch this, output "test result must be between 0 and 100" and allow the user another input.		
13	To test the validation of the test result input. with an extreme high input.	"99999"	For the validation to catch this, output "test result must be between 0 and 100" and allow the user another input		
14	To test the functionality of the "Distinction alert"	"Bob Smith" Inputted as the Students Name "72" inputted as the test result	For the if statement following the test result input to catch this, output "Distinction has been achieved for Bob Smith" for the grade "Distinction" to then be stored to the list_grade		
15	To test the functionality of the "Storing a merit result"	"62" inputted as the test result	For the elif statement to catch this and store "merit" to list_grade		
16	To test the functionality of the "Storing a pass result"	"42" inputted as the test result	For the elif statement to catch this and store pass" to list_grade		
17	To test the functionality of the "Storing a fail result"	"35" inputted as the test result	For the elif statement to catch nothing and instead result to "else", resulting to fail being stored to list_grade		



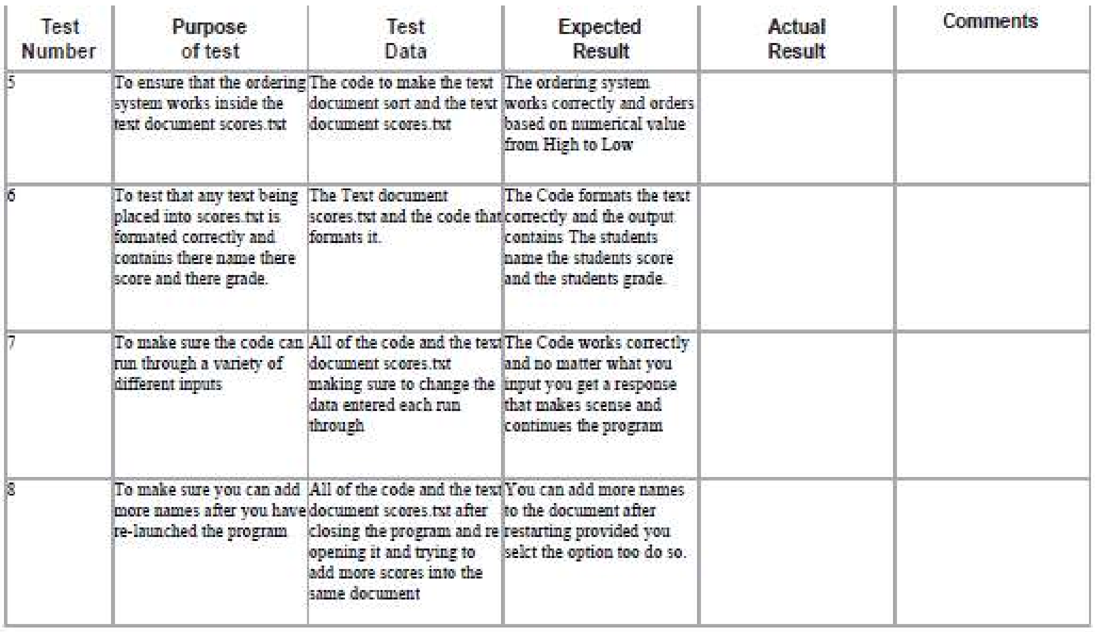
Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
18	To test if the loop is broken once recorded students exceeds number_of_students	6 inputted as number of students, To input the data 6 times	Loop to be broken allowing the following lines of code to be executed.		
19	To test if the sort functions as intended, from highest to lowest in terms of test results.	"55, 67, 34, 88, 64, 99" Inputted as 6 separate test results	To be ordered "34,55,64,57,88,99" after the sort has been executed.		
20	To test that the results correctly write to a text file.	For the following data to be inputted throughout the program; "Bob Smith, 55, Merit" "Jack Jones, 42, Pass" "Luke Daniels, 55, merit" "Charlie boyd, 80, Distinction" "Harry Brown, 45, Pass" "Rachel Robinson, 55, Merit"	For the data to be correctly written to the text file, The following data should be present when the text file is read. ""Bob Smith, 55, Merit" "Jack Jones, 42, Pass" "Luke Daniels, 55, merit" "Charlie boyd, 80, Distinction" "Harry Brown, 45, Pass" "Rachel Robinson, 55, Merit""		

Lead Examiner Commentary:

The learner has produced a thorough test plan to confirm a working solution which includes a range of data. Expected results are specific and accurate based on identified test data. The level of detail is not quite enough for the top of the mark band.
Mark in **band 3 (5 marks)**.



Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
1	Test that the program launches correctly	Visual confomation of the program running. It will test that the program is stable and is able to be run.	The program will lauch correctly and allow me to answer the first question my program asks the user		
2	Test if the question about having more than 6 students in your class works as intended and outputs correctly based on your answer	The code related to having above 6 students in your class and the code that gives responces to your input	If you say you have above 6 students it will allow you to continue and if you have below 6 it will display a message telling you that you need atleast 6 Students.		
3	To make sure the program creates the nessasary scores.txt file	The file and code that creates the file will need to be looked at simply running the code should create the file	When i run the code a file will be created named scores.txt		
4	To ensure the grading works correctly and assigns the correct grade to the apropriate score.	The outputed text into the text document scores.txt and the code to assign a grade to a score	The code will assign the correct grade to the apropriate scors and store them in the text document scores.txt		



Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
5	To ensure that the ordering system works inside the text document scores.txt	The code to make the text document sort and the text document scores.txt	The ordering system works correctly and orders based on numerical value from High to Low		
6	To test that any text being placed into scores.txt is formated correctly and contains there name there score and there grade.	The Text document scores.txt and the code that formats it.	The Code formats the text correctly and the output contains The students name the students score and the students grade.		
7	To make sure the code can run through a variety of different inputs	All of the code and the text document scores.txt making sure to change the data entered each run through	The Code works correctly and no matter what you input you get a response that makes scense and continues the program		
8	To make sure you can add more names after you have re-launched the program	All of the code and the text document scores.txt after closing the program and re opening it and trying to add more scores into the same document	You can add more names to the document after restarting provided you selct the option too do so.		

Lead Examiner Commentary:

The plan shows what needs testing however is not adequate as no test data is provided. The plan refers to functional items but does not describe how it is to be tested.
Mark in **band 1 (1 mark)**.

Activity 4 – Code

```
#Junior Software Development Project

StuN=[]
StuR=[]
StuG=[]


#Input a number of students no less than 6
while True:
    NoS=int(input("Please enter the total number of students. No less than 6:
"))

    if  NoS < 6:
       print("There cannot be less than 6 students in the class")
    else:
       break;

#enter name and marks for student
StuG=["null"]*NoS

for a in range (NoS):
   print("Student number",a+1)
   x=input("Please give the name of the this student")
   StuN.append(x)
   while True:
      y=int(input("Please enter the marks this student recieved"))
      if y>=0 and y<=100:
         break
   StuR.append(y)

#Create the StuG

for a in range (NoS):
   if StuR[a]<=40:
      StuG[a]="F"
   elif StuR[a]<=50:
      StuG[a]="P"
   elif StuR[a]<=69:
      StuG[a]="M"
   else:
      StuG[a]="Distinction"
print()
print("Before the bubble sort...")
print(StuN)
print(StuR)
print(StuG)

#Bubble sort - descending order according to the StuG
def bubblesort():
   for a in range(0, len(StuR) - 1 ) :
      for i in range(0, len(StuR) - 1) :
        if StuR[i] < StuR[i + 1] :
            tmp = StuR[i]
            StuR[i] = StuR[i + 1]
            StuR[i + 1] = tmp

            tmp = StuN[i]
            StuN[i] = StuN[i + 1]
            StuN[i + 1] = tmp

            tmp = StuG[i]
            StuG[i] = StuG[i + 1]
            StuG[i + 1] = tmp
bubblesort()

print()
print("After the bubble sort...")
print(StuN)
print(StuR)
print(StuG)

#Calculate if there are any Distinctions
print()
count=0
for a in range (NoS):
    if str(StuG[a])=="Distinction":
        count=count+1
#If count remains 0 that means there were no distinctions

for a in range (NoS):
    print(StuN[a], " ",StuR[a])

if count==0:
    print("No Distinctions were achieved")
else:
    print("\n"+"There were ", count,  " Distinctions by the following students:")
    #linear search
    for a in range (NoS):
        if str(StuG[a])=="Distinction":
           print(StuN[a])


#we are going to output in a file the test StuR in order and if there are any
distinctions
print()
f=open("StuR.txt"
       , "a")
f.write("StuR:")
for a in range (NoS):
    f.write("\n"+StuN[a]+": "+str(StuR[a])+" | "+str(StuG[a]))
f.close()
```

Lead Examiner Commentary:

The learner has produced a program that fully meets all the requirements. Accurate syntax and indentation have been used throughout the code and commenting is consistently clear and informative. Program outputs are accurate and informative, validation and other checks have been used which are all accurate, resulting in a largely robust program being created.

Mark in **band 4 (20 marks)**.

```
#The line below should create a text file names "scores.txt"
sf = open("scores.txt", "w+")


#The lines below should check if you have the correct ammount of
students in the class
students = int(input("How many students do you have in your class? "))
if students < 6:
    print("You need atleast 6 students!")
 #If they have the correct ammont of students they are asked to enter
their name
else:
    name = input("Enter students name: ")
    print (name)
    #They are then asked to enter the students score and it then prints
name and score
    score = int(input("Enter the students score: "))
    #Defining two inputs can make it eaisier to input the data to text file and
makes it easier to sort
    ns = (name, score)
    print(ns)
correct = input("Is this infomation correct?")
if correct == "yes":
    print("ok")
else:
    exit


#This checks to see if the score is under 40 for a fail
if score < 40:
    grade = "fail"
    print("The students grade is:", grade)
#This checks to see if their grade is a pass
```

```
if 40 < score < 51:
   grade = "pass"
#This checks to see if their grade is a merit
if 51 < score < 69:
   grade = "merit"
#And this checks to see if their grade is a distinction
if score > 70:
   grade = "distinction"
print("The students grade is: ", grade)
#This will open the file scores.txt using the 'a' so python knows im trying to
ammend a already created file
open('scores.txt', 'w', 1)

print(grade, file=sf)
print("")
print(ns, grade)


   #f = open("filename.txt", "+w")
   #f.open ("What you want to type", "+a")
   #sf.write("")
```

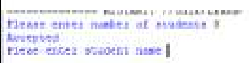
Lead Examiner Commentary:

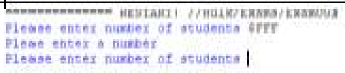
The learner has produced a program that meets some of the requirements. There are errors in the logic and the program is not complete. The logic is imprecise and there are errors in the outcomes. The only processes one student, although the code generates the file no data is written to it.

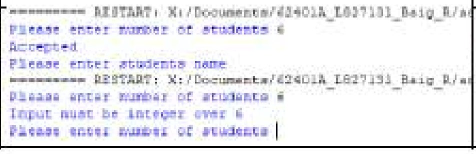
Mark in **band 1 (6 marks).**

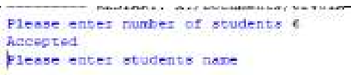
Activity 4 - Testing

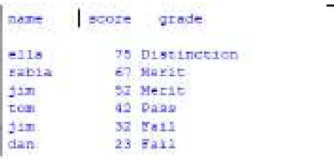
TEST NUMBER	PURPOSE OF TEST	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	COMMENTS

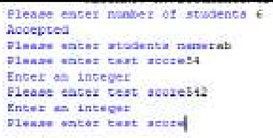
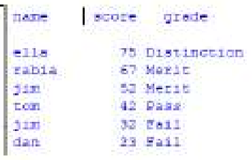
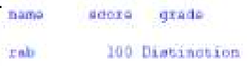
1	Test if the program accepted valid input for the number of students	8	The program should accept a valid input for number of students so anything about 6 would be acceptable	
				`============== RESTART: //[illegible]`
				`Please enter number of students 8`
				`Accepted`
				`Plese enter student name`
					This shows that it accepts the data

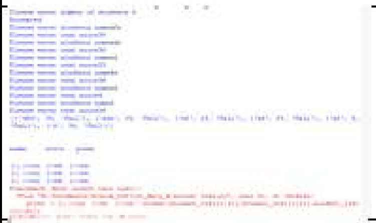
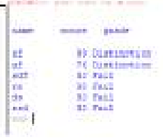
2	Tests the program so it does not accept abnormal data	$FFF	The program should not accept data that isn't an integer	
				`=============== RESTART: //[illegible]`
				`Please enter number of students $FFF`
				`Plese enter a number`
				`Please enter number of students`
					This shows it doesnt accept abnormal data and if its entered it will repeat the question again

3	Test the program so that it accepts the lower bound (minimum number of students which is 6)	6	The program should accept the lower bound for the number of students which is 6	
				`========== RESTART: X:/Documents/424O1A_L027131_Baig_R/a`
				`Please enter number of students 6`
				`Accepted`
				`Please enter students name`
				`========== RESTART: X:/Documents/424O1A_L027131_Baig_R/a`
				`Please enter number of students 5`
				`Input must be integer over 6`
				`Please enter number of students`
					There was an error in my code, the variable was wrong and didn't match with the rest of the code so it wouldn't allow me to enter the lower bound

4	Testing the program and making sure the variable is fixed so it will accept the lower bound	6	The program should accept the lower bound which is 6	
				`Please enter number of students 6`
				`Accepted`
				`Please enter students name`
					I changed the variable name which allowed the code to accept the lower bound

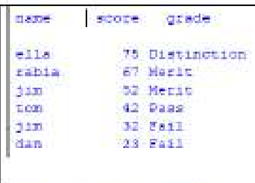
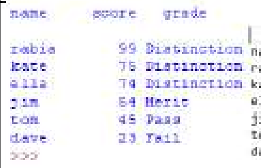
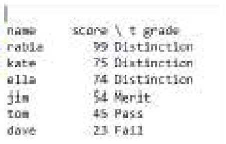
5	Tests the program so it accepts a valid input for the students name	Ella rabia etc	The program should accept the students name which is a string	
				`name     score   grade`
				
				`ella        75 Distinction`
				`rabia       67 Merit`
				`jim         52 Merit`
				`tom         42 Pass`
				`jim         32 Fail`
				`dan         23 Fail`
					It outputs the name correctly and accepts strings

6	Tests the program so it does not accept abnormal data for the students name	6	The program should not accept abnormal data for example integers shouldn't be accepted	 Please enter number of students 6 Accepted Please enter students namerab Please enter test score54 Enter an integer Please enter test score542 Enter an integer Please enter test score	Im having difficulties with the variable score and to make sure it doesn't accept data that is extreme or abnormal
7	Test the program so it accepts valid data for the tests score		The program should accept valid data for the test score so any integer below 100 should be accepted	 name score grade elle 75 Distinction rabia 67 Merit jim 52 Merit tom 42 Pass jim 32 Fail dan 23 Fail	Valid data is accepted and works
8	Tests the program so it does not accept abnormal data for the test score		The program should accept data that isn't an integer		
9	Test the program so it accepts the higher bound for the test score which is 100	100	The program should accept the highest bound for the scoretest which is 100	 name score grade rab 100 Distinction	It accepts 100
10	Tests the program so it doesn't accept abnormal data for test score so anything above 100		The program shouldn't accept integers that are above 100		
11	Tests the program so				

	the grade boundaries are met				
12	Test so it displays the graph correctly		The graph will be presented from highest to lowest		The graph isn't presented properly so I need to go back to my code to resolve it
13	Test code so graph displaying grades is shown		The graph should present the grades from highest to lowest		It displays the grades correctly from highest to lowest

```
[illegible]
```

```
name     score    grade

[illegible]  83 Distinction
[illegible]  76 Distinction
[illegible]  32 Fail
[illegible]  32 Fail
[illegible]  32 Fail
[illegible]  32 Fail
>>> |
```

14	The program should follow the grading requirements	75 67 52 42 32 23			It follows the grading requirements and matches the teachers mark scheme
15	The program should write the test results to a text file		It should output the results onto a text file	 	The text file works successfully and outputs the results onto notepad correctly.

```
name      score   grade

ella         75 Distinction
rabia        67 Merit
jim          52 Merit
tom          42 Pass
jim          32 Fail
dan          23 Fail
```

```
name        score   grade

rabia         99 Distinction
kate          75 Distinction
ella          74 Distinction
jim           64 Merit
tom           45 Pass
dave          23 Fail
>>>
```

```
name      score \ t grade
rabia         99 Distinction
kate          75 Distinction
ella          74 Distinction
jim           54 Merit
tom           45 Pass
dave          23 Fail
```

Lead Examiner Commentary:

The learner has produced some evidence of an iterative development process that identifies and resolves some basic errors. Comments show understanding of the basic errors and how they were fixed.

Mark in **band 2 (3 marks).**

Pearson BTEC

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
1	To test the functionality of the number of students input.	"6"	For the program to continue with no issues, assigning the value of 6 to the number of students.	Program continues as expected with no issues, successfully assigns the value of 6 to the number_of_students variable.	Works as intended
2	To test the validation of the number of students input with an anomaly input.	"a"	For the validation to catch this, output "Input invalid" and allow the user another input.	Program outputs "Input invalid!, please try again" and allows the user to enter another value	Works as intended This will repeat until a valid input is inputted.
3	To test the validation of the number of students input with an extreme low input.	"-10"	For the validation to catch this, output "Input invalid" and allow the user another input.	Program outputs "minimum of 6 students required for a class", and allows the user to enter another value	The program doesn't realise this is an extreme low however understands there must be a minimum of 6. This is sufficient validation.
4	To test the validation of the number of students input with an extreme high input.	"9999999999999999999 9999999999999999999 999999"	For the validation to catch this, output "Input invalid" and allow the user another input.	Program continue as normally, assigning the test data to the number of students.	The program can not determine that this is an unrealistic number of students in a class.
5	To test the minimum number of student's validation.	5	For the validation to catch this, output "minimum of 6 students required for a class" and allow the user to input another value.	Program outputs "output "minimum of 6 students required for a class" and allow the user to input another value."	Works as intended

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
6	To test the functionality of the student's name input.	"Bob Smith"	For the program to continue with no issues, assigning the name of "bob smith" to the variable of student_name.		
7	To test the validation of the student's name input. With an anomaly input.	"1"	For the validation to catch this, output "Input invalid" and allow the user another input.		
8	To test the validation of the student's name input. with an extreme low input.	"a"	For the validation to catch this, output "Input invalid" and allow the user another input.		
9	To test the validation of the student's name input. with an extreme high input.	"aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaa"	For the validation to catch this, output "Input invalid" and allow the user another input.		
10	To test the functionality of the test result input.	"48"	For the program to continue with no issues, assigning the test result of "48" to the variable of test result.		

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
11	To test the validation of the test result input. With an anomaly input.	"a"	For the validation to catch this, output "Input invalid" and allow the user another input.		
12	To test the validation of the test result input. with an extreme low input.	"-100"	For the validation to catch this, output "test result must be between 0 and 100" and allow the user another input.		
13	To test the validation of the test result input. with an extreme high input.	"99999"	For the validation to catch this, output "test result must be between 0 and 100" and allow the user another input		
14	To test the functionality of the "Distinction alert"	"Bob Smith" Inputted as the Students Name "72" inputted as the test result	For the if statement following the test result input to catch this, output "Distinction has been achieved for Bob Smith" for the grade "Distinction" to then be stored to the list_grade		
15	To test the functionality of the "Storing a merit result"	"62" inputted as the test result	For the elif statement to catch this and store "merit" to list_grade		
16	To test the functionality of the "Storing a pass result"	"42" inputted as the test result	For the elif statement to catch this and store pass" to list_grade		
17	To test the functionality of the "Storing a fail result"	"35" inputted as the test result	For the elif statement to catch nothing and instead result to "else", resulting to fail being stored to list_grade		
18	To test if the loop is broken once recorded students exceeds number_of_students	6 inputted as number of students, To input the data 6 times	Loop to be broken allowing the following lines of code to be executed.		
19	To test if the sort functions as intended, from highest to lowest in terms of test results.	"55, 67, 34, 88, 64, 99" Inputted as 6 separate test results	To be ordered "34,55,64,57,88,99" after the sort has been executed.		

Lead Examiner Commentary:

Testing shows evidence of a limited or linear development process, with minimal identification and resolution of errors. To get into mark band 2 there must be evidence of errors and how they have been solved.

Mark in **band 1 (2 marks)**.

Activity 5

Evaluation

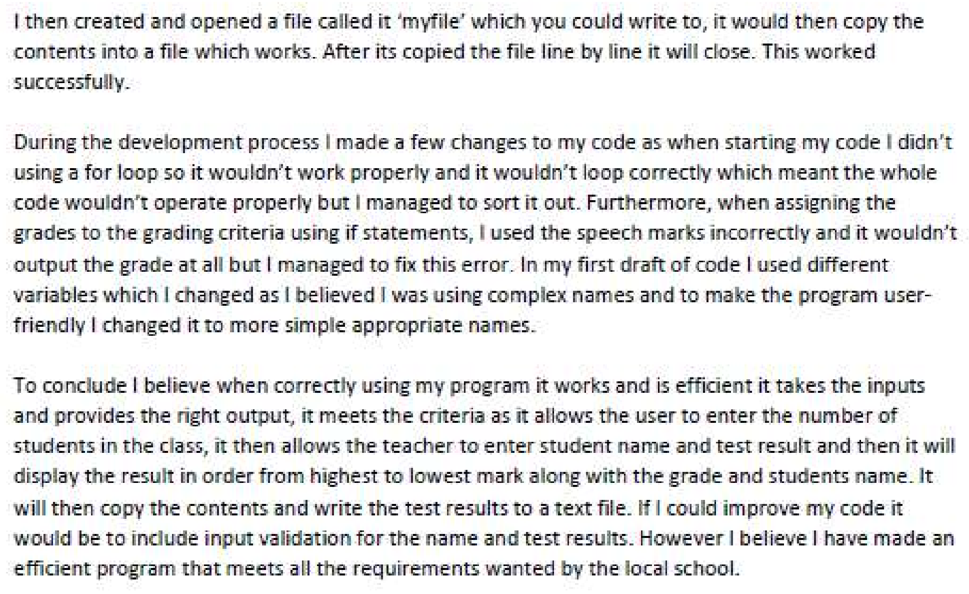
I was employed by a local school to create a program for teachers that allow them to record test results for students and display them in order from highest to lowest.

The program asks the user to enter the number of students, it will then ask for the user's name and score results the same amount of times the amount of students were entered. Once the user has entered all the details it will then present a graph which will present the test results from highest to lowest along with the grade and student's name.

When the user enters the amount of students, I created a while loop to ensure that the it meets the requirements for the minimum amount of students which is 6 if the user enters 6 or above, it will then carry on to the next section of code, if the user enters a number smaller than 6, it will repeat the question until the user inputs a valid number. I made sure that I used integer and not a float as the code is only accepting whole numbers.
I created an empty list so then when I have name, score and grade I can add it to the list using .append
After the list I decided to use a for loop so that it repeats the number of times the number of students were entered to make the code easier and simpler to understand. I then created two variables called score and name which allows the user to enter these details. The score was an integer and the name was a string. After I used if statements as in the requirements there was a mark scheme so if the score was less than 40 it will be a fail, if the score was between 41 and 50 it was a pass, if the score was between 51-69 it would be a merit and if it was 70 and above it would be a distinction.
The score, name and grade would then be added to the list. I then created the graph so it would list the users name, score and grade from highest to lowest.

In terms of quality and performance I believe my solution is efficient to an extent, as I did struggle to use input validation for the name and score entered inside the for loop, so if a user enters a number in the name it wouldn't loop and ask again until entered properly similar to the score entered it wouldn't loop. I believe I have used suitable data types and named them appropriately to make it easier, I believe my code is user-friendly by adding spacing to the end of each question and in the graph so users can clearly identify whose grade is whose.

The choices I made for my coding conventions was so that it was easy to read and follow and easy for a novice user to understand my code, I made it simple as possible by using suitable variables and while loops for input validation. I created an empty list at the start of the code so that then I can call it later on and add to the list by using .append. I declared functions so that I can reuse that block of code later on as its reusable making the code look more organized and less jumbled. I used a for loop so that it would it would loop the same amount of times as number of students entered. I then printed the list so it would output the results onto the screen, the results were outputted in order of the user entering them so it wasn't highest to lowest.
When presenting the results in the graph I then sorted the list to make sure it would output the results from highest to lowest by using .sort (reverse = True). I made the graph easy to read and follow by using appropriate spacing and easy headings eg name, mark, grade


I then created and opened a file called it 'myfile' which you could write to, it would then copy the contents into a file which works. After its copied the file line by line it will close. This worked successfully.

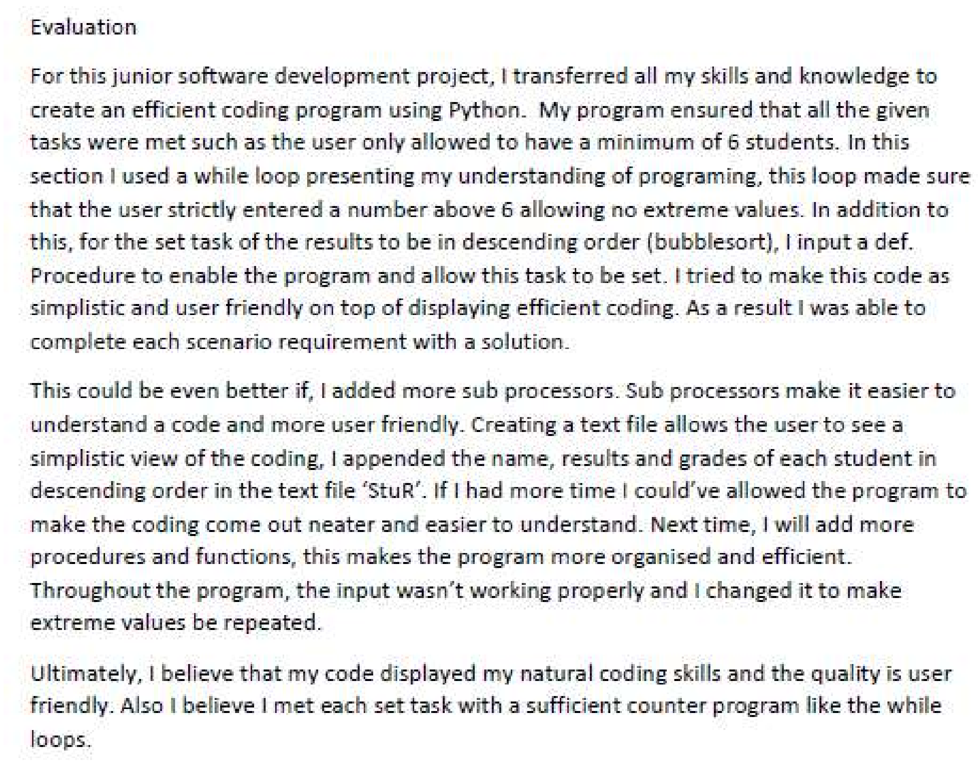
During the development process I made a few changes to my code as when starting my code I didn't using a for loop so it wouldn't work properly and it wouldn't loop correctly which meant the whole code wouldn't operate properly but I managed to sort it out. Furthermore, when assigning the grades to the grading criteria using if statements, I used the speech marks incorrectly and it wouldn't output the grade at all but I managed to fix this error. In my first draft of code I used different variables which I changed as I believed I was using complex names and to make the program user-friendly I changed it to more simple appropriate names.

To conclude I believe when correctly using my program it works and is efficient it takes the inputs and provides the right output, it meets the criteria as it allows the user to enter the number of students in the class, it then allows the teacher to enter student name and test result and then it will display the result in order from highest to lowest mark along with the grade and students name. It will then copy the contents and write the test results to a text file. If I could improve my code it would be to include input validation for the name and test results. However I believe I have made an efficient program that meets all the requirements wanted by the local school.

Lead Examiner Commentary:

The learner has demonstrated a mostly accurate and detailed understanding of technical concepts. Valid and mostly supported justification of coding conventions used, and the learner has made logical links between aspects of the solution and the requirements of the scenario. Valid and mostly supported judgements of the quality and performance of the program. Accurate technical vocabulary used to support arguments.

Mark in **band 3** (**9 marks**).


Evaluation

For this junior software development project, I transferred all my skills and knowledge to create an efficient coding program using Python. My program ensured that all the given tasks were met such as the user only allowed to have a minimum of 6 students. In this section I used a while loop presenting my understanding of programing, this loop made sure that the user strictly entered a number above 6 allowing no extreme values. In addition to this, for the set task of the results to be in descending order (bubblesort), I input a def. Procedure to enable the program and allow this task to be set. I tried to make this code as simplistic and user friendly on top of displaying efficient coding. As a result I was able to complete each scenario requirement with a solution.

This could be even better if, I added more sub processors. Sub processors make it easier to understand a code and more user friendly. Creating a text file allows the user to see a simplistic view of the coding, I appended the name, results and grades of each student in descending order in the text file 'StuR'. If I had more time I could've allowed the program to make the coding come out neater and easier to understand. Next time, I will add more procedures and functions, this makes the program more organised and efficient. Throughout the program, the input wasn't working properly and I changed it to make extreme values be repeated.

Ultimately, I believe that my code displayed my natural coding skills and the quality is user friendly. Also I believe I met each set task with a sufficient counter program like the while loops.

Lead Examiner Commentary:

The learner has demonstrated superficial understanding of relevant technical concepts. There is unsupported justification of changes made during the development process and limited justification of coding conventions supported. Limited judgements about the quality and performance of the program keeps this evaluation in **mark band 1 (3 marks)**.

Summary

Based on performance in this examination series, learners are offered the following advice:

- Apply their knowledge to as many different scenarios as possible. The exam paper will always contain 5 activities which always be the same just the scenario would be different and therefore this will prepare learners to be able to provide answers to the given context under exam conditions.
- Use standard naming conventions throughout the design process and clearly demonstrate this in the flowchart and pseudocode.
- Pseudocode needs to be a detailed yet readable description of what a computer program must do, expressed in a natural language rather than in a programming language if top marks are to be achieved.
- Develop a better understanding of the testing process. Test plan must include normal, abnormal and extreme data. Testing must address errors encountered and how these were overcome. The testing must be iterative, document tests when code is being developed as this will give a true reflection of the development.
- Ensure the Program uses accurate validation and error checking procedures throughout, resulting in a robust program that minimises errors and handles unexpected events. This will enhance the completed solution and allow the higher mark bands to be accessed. Programs must address most requirements to gain higher marks.
- The evaluation needs to include a fully supported justification of changes made during the development process, as well as a fully supported justification of coding conventions selected if higher mark bands are to be accessed.