L3 Lead Examiner Report 1901

January 2019

L3 Qualification in Computing

Unit 4: Software Design and Development Project

Edexcel and BTEC Qualifications

Edexcel and BTEC qualifications come from Pearson, the world's leading learning company. We provide a wide range of qualifications including academic, vocational, occupational and specific programmes for employers. For further information visit our qualifications website at http://qualifications.pearson.com/en/home.html for our BTEC qualifications.

Alternatively, you can get in touch with us using the details on our contact us page at http://qualifications.pearson.com/en/contact-us.html

If you have any subject specific questions about this specification that require the help of a subject specialist, you can speak directly to the subject team at Pearson. Their contact details can be found on this link:
http://qualifications.pearson.com/en/support/support-for-you/teachers.html

You can also use our online Ask the Expert service at https://www.edexcelonline.com
You will need an Edexcel Online username and password to access this service.

Pearson: helping people progress, everywhere

Our aim is to help everyone progress in their lives through education. We believe in every kind of learning, for all kinds of people, wherever they are in the world. We've been involved in education for over 150 years, and by working across 70 countries, in 100 languages, we have built an international reputation for our commitment to high standards and raising achievement through innovation in education. Find out more about how we can help you and your learners at: www.pearson.com/uk

January 2019
Publications Code 31771H_1901_ER

Grade Boundaries

What is a grade boundary?

A grade boundary is where we set the level of achievement required to obtain a certain grade for the externally assessed unit. We set grade boundaries for each grade, at Distinction, Merit and Pass.

Setting grade boundaries

When we set grade boundaries, we look at the performance of every learner who took the external assessment. When we can see the full picture of performance, our experts are then able to decide where best to place the grade boundaries – this means that they decide what the lowest possible mark is for a particular grade.

When our experts set the grade boundaries, they make sure that learners receive grades which reflect their ability. Awarding grade boundaries is conducted to ensure learners achieve the grade they deserve to achieve, irrespective of variation in the external assessment.

Variations in external assessments

Each external assessment we set asks different questions and may assess different parts of the unit content outlined in the specification. It would be unfair to learners if we set the same grade boundaries for each assessment, because then it would not take accessibility into account.

Grade boundaries for this, and all other papers, are on the website via this link:

http://qualifications.pearson.com/en/support/support-topics/results-certification/grade-boundaries.html

Unit 4: Software Design and Development Project

Grade	Unclassified	Level 3		
		P	M	D

Boundary Mark	**0**	**24**	**37**	**50**

Introduction

This was the second examination season for Level 3 BTEC Computing Unit 3: 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 slightly better compared to the previous season for this unit. It was evident that some learners were well prepared for the rigour of this assessment.

The performance on Activity 1 performed as expected with many learners picking up marks in band 2. Most of the responses used BCS symbols and had a good go at breaking down the requirements into relevant parts.

Activity 2 was of a good standard and demonstrated the learner ability to apply pseudocode design methodologies to a scenario. Learners have taken on board previous comments regarding this activity and the number of pseudocode being too close to the coding was much less compared to June 2018.

Activity 3 & 4 (testing) was poor again this series and resulted in most 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 process. In most cases, the testing carried out did not evidence any errors encountered which is essential for accessing higher marks.

Activity 4 (Coding) was done to a good standard by the learners. Some were awarded full marks as they produced a working solution along with detailed comments.

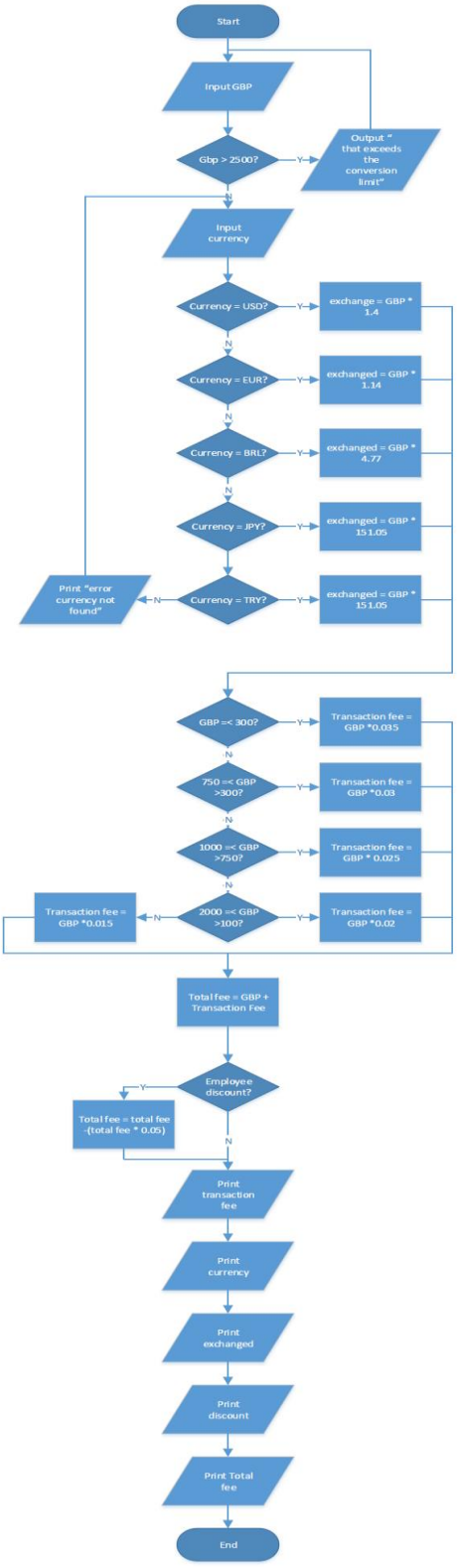
The evaluations (activity 5) were of a good standard and most learners accessed bands two and three. Some learners only produced a review of what they did which resulted in marks from band 1 being awarded.

Individual Questions

Tests or Exams

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.

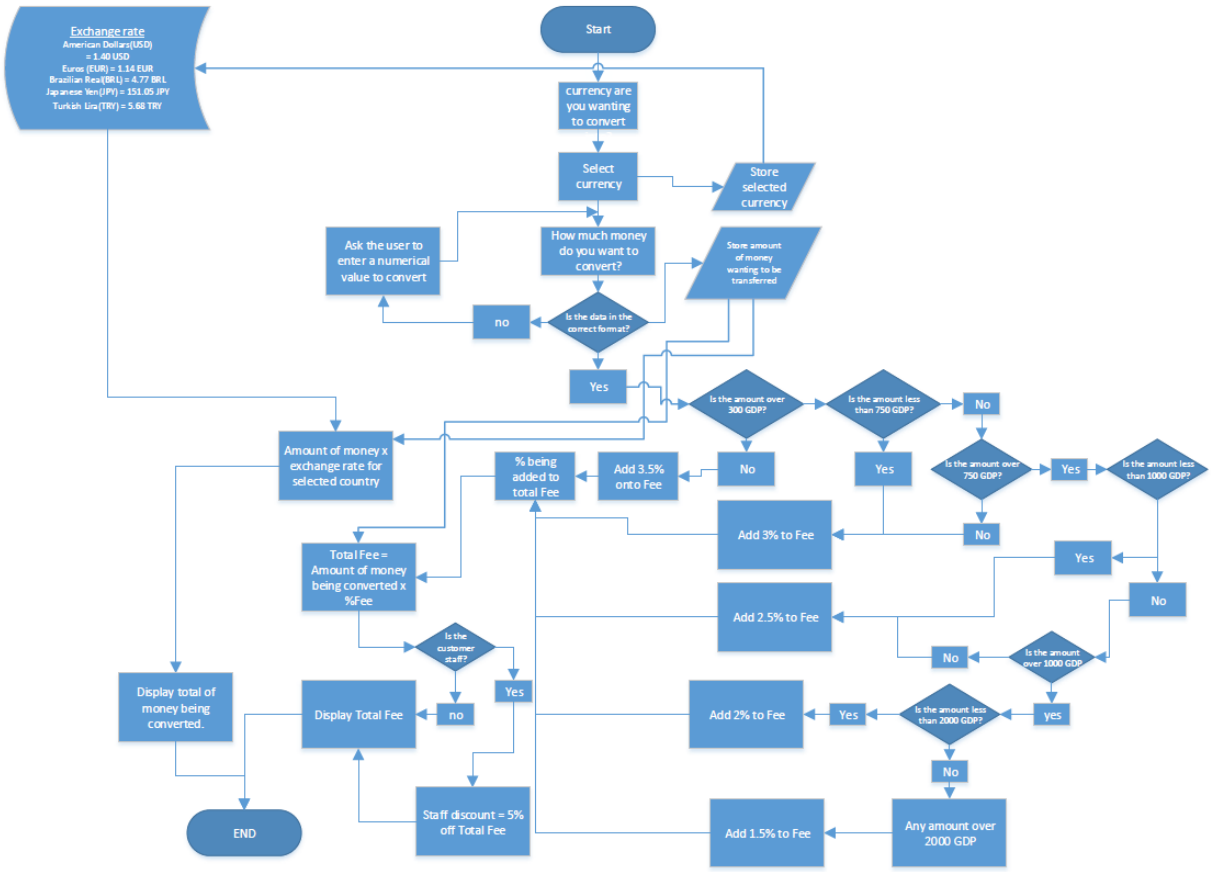
Activity 1


Start
Input GBP
Gbp > 2500?
Output " that exceeds the conversion limit"
Input currency
Currency = USD?
exchange = GBP * 1.4
Currency = EUR?
exchanged = GBP * 1.14
Currency = BRL?
exchanged = GBP * 4.77
Currency = JPY?
exchanged = GBP * 151.05
Currency = TRY?
exchanged = GBP * 151.05
Print "error currency not found"
GBP =< 300?
Transaction fee = GBP *0.035
750 =< GBP >300?
Transaction fee = GBP *0.03
1000 =< GBP >750?
Transaction fee = GBP * 0.025
2000 =< GBP >100?
Transaction fee = GBP *0.02
Transaction fee = GBP *0.015
Total fee = GBP + Transaction Fee
Employee discount?
Total fee = total fee -(total fee * 0.05)
Print transaction fee
Print currency
Print exchanged
Print discount
Print Total fee
End

The learner has had a good go at the flow chart for the specified problem.

British Computer Society (BCS) flowchart symbols have been used accurately throughout as well as the breaking down of requirements into component parts that are detailed and relevant.

The flowchart shows full coverage of inputs, outputs and processes using naming conventions appropriate to the scenario consistently. Although some logical errors have been identified it is still deserving of a mark in **band 3 (8 marks).**



The learner has addressed all aspects of the flow chart for the specified problem. British Computer Society (BCS) flowchart symbols have been used but are mostly incorrect with irrelevant parts. There is some evidence of breaking down of requirements into component parts that are relevant.

Links between component parts are incomplete with limited procedures for handling unexpected events.

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

Activity 2

```
Begin
        Print (“welcome”)
        While w = True:
                Start = input ("do you wish to begin?")
                 If Start = "YES":
                        w = False
                Else if Start = "NO":
                         Print ("ok")
                        Print ("GOODBYE!")
                        END
                Else print ("input error try again")
        While x = true:
                Try:
                        GBP = input (“please enter the amount in pounds to wish to exchange”)
                        If GBP > 2500
                                Print (“that exceeds the legal conversion limit”)
                        Else x = false
                        End if
                Except value Error
                        Print (“error try again”)

        While y = true
                Currency = input (“please enter the currency you wish to convert to”)
                If Currency. Upper = “USD” then
                        Exchanged = GBP * 1.4
                        y = false
                Elseif Currency = “EUR” then
                        Exchanged = GBP * 1.14
                        y = false
                Elseif Currency = “BRL” then
                        Exchanged = GBP * 4.77
```

```
            y = false
        Elseif Currency = "JPY" then
            Exchanged = GBP * 151.05
            y = false
        Elseif Currency.upper = "TRY" then
            Exchanged = GBP * 5.68
            y = false
        Else print ("I'm sorry that currency either does not exist or we do not exchange it")
        End if

If GBP =< 300 then
        TransactionFee = GBP * 0.035
Elseif GBP >300 AND GBP =< 750 then
        TransactionFee = GBP * 0.03
Elseif GBP > 750 AND GBP =< 100 then
        TransactionFee = GBP * 0.025
Elseif GBP >1000 AND GBP =< 2000 then
        TransactionFee = GBP * 0.02
Else TransactionFee = GBP * 0.015
End if

Totalfee = GBP + TransactionFee

If employee discount = true
Disountmoney = totalfee * 0.05
Discounted= Totalfee – discountmoney
End if

Print ("you will recieve" + Exchanged + "IN" + Currency)
Print ("the fee for this transaction is" + TransactionFee)
Print ("total fee" + TotalFee)
```

```
        If employeediscount = true
                Print ("you are eligible for 5% off making your total" + discounted)
        endif

END
```

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 (8 marks)**.

START

Exchange_Rate [[USD, EUR, BRL, JPY, TRY] [1.40, 1.14, 4.77, 151.05, 5.68]]

PRINT (“How much money do you wish to convert?”)

Amount = int(input(“ “)

PRINT (“Please indicate which currency you wish to convert to by using either USD (American Dollars), EUR (Euros), BRL (Brazilian Real), JPY (Japanese Yen), TRY (Turkish Lira).”)

Selected_Currency = str(input(“ ”))

WHILE Selected_Currency != values contained within Exchange_Rate :

THEN print (“ Please use the indicated values to show which currency you wish to convert into”)

IF Selected_Currency == value contained within Exchange_Rate :

THEN multiply Amount by Exchange_Rate

THAT = Exchange_Value

PRINT (Exchange_value)

BREAK

IF Amount =< 300 :

THEN multiply Amount by 3.5%

THAT = Total_Fee

ELIF Amount =< 750 :

THEN multiply Amount by 3%

THAT = Total_Fee

ELIF Amount =< 1000 :

THEN multiply Amount by 2.5%

THAT = Total_Fee

ELIF Amount =< 2000

THEN multiply Amount by 2%

THAT = Total_Fee

ELSE Amount >= 2000 :

THEN multiply Amount by 1.5%

THAT = Total_Fee

```
PRINT ("do you have a staff discount? Please indicate using a Y or N")
IF INPUT = Y:
        THEN Final_FEE = Total_Fee minus 5%
ELSE Final_Fee == Total_Fee

PRINT (Final_FEE)
END
```

The learner has produced a structure which shows appropriate and consistent use of hierarchy and indentation, providing clarity and mostly readable pseudocode. However, the pseudocode will not provide a working solution.

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

Activity 3

Test Plan

Test #	Purpose	Data	Expected outcome	actual outcome	Comments
#1	to check that a user cannot enter a value of more than 2500 when entering GBP	entered "90000"	the computer will say that it exceeds the limit and loop back to the input for the GBP		
#2	To check that the loop does not last infinitely when an appropriate value has been entered	entered "100"	the computer shall move on to the next input of the intended foreign country		
#3	To test the input for currency. Firstly to distinguish between a real and made up currency, then to check its loop works appropriately	entered "AAA"	the programme shows an error that it doesn't recognise the input and then loops back to the prior input menu		
#4	to test the employee discount if statements loops correctly	entered "a"	the programme shows an error that it doesn't recognise the input and then loops back to the prior input menu		
#5	to test the employee discount if statements loops correctly	entered "y" for discount and 1000 to USD for exchange	the programme subtracts 5% from the total amount and displays it at the end along with the pre-discount total		
#6	N test for employee discount	entered "n" for the discount and 1000 to USD for exchange	the post and pre discount amount should be the same (1025)		
#7	checking that all currencies are calculated properly	entered 1000 into every available foreign currency	1400 in USD, 1140 in EUR, 4770 in BRL, 151050 in JPY and 5680 in TRY		
#8	Making sure the receipt works properly	100 usd not an employee	TRANSACTION RECIEPT POUNDS GIVEN 100 USD RECIEVED 140.0		

			TRANSACTION FEE 3.5000000000000004 TOTAL COST 103.5 APPLICABLE DISCOUNT 0 FINAL PAYABLE 103.5 will be displayed on a text file opened at the end of the program		
#9	Making sure the try except works for the type error in the GBP input	Inputting "aaaaa" into the GBP variable input	Error plese enter an amount in whole pounds will be displayed on screen		
#10	Date time check	n/a	After the welcome has displayed the time will be displayed under it		
#11	Does the receipt file match the print	100, yen, yes an employee	Receipt and programme will say the same thing however in different formats		
#12	Repetition checking	-	At the endThe programme will repeat without any information from the previous go		

The learner has produced an adequate test plan to confirm a working solution which includes a range of data. Expected results are specific and accurate based in the test data.

Mark in **band 2 (4 marks)**.

Document for Activities 3 and 4

Test Plan (add additional rows as required)

Program language the product is to be produced in (tick box for language used):

Python ☐ C Family ☐

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
1	To test if a user can enter a number to convert.		No error message should be produced		
2	Test if an error is created when a non-numerical value is entered		An error should appear asking for a number.		
3	To test if the program accepts A, B, C, D, E for selection of conversion currency.		The program should accept A, B, C, D, and E as valid data for selection of conversion currency.		
4	To test if the program produces an error message if any other values are entered for the selection of conversion currency. The error message should loop until the correct data is entered.		The program should produce an error message asking for the user to enter a valid data form of data. The message should loop until a valid entry is made.		
5	To test if the program produces an error message if a numerical data type is entered. This error message should loop until the correct data type is entered.		An error message should be produced asking for the user to input a valid data type. This message should loop until a valid data type is entered.		
6	To test if the user is able to enter a numerical value for how much money they wish to convert.		The user should be able to enter a numerical value for the amount they wish to convert.		
7	To test if an error message is produced when a non-numerical value is entered.		An error message should appear.		

The learner has produced a test plan but has no test data. Expected results are generic, no marks can be awarded for this test plan.

Mark in **band 0 (0 marks)**.

Activity 4 - Program

```
import time
import os                                                    #these python imports will
allow for extra functionality
from datetime import datetime
import random


V = True                                                     #this variable allows the
whole program to loop until the user quits.


def loading():                                               #fake loading messsage
serves no real purpose but breaks up the flurry of infor and looks neat
    loadtime = random.randint(3,10)
    print("CALCULATING")
    for loadtime in range(loadtime):
        print(".")
        time.sleep(0.3)


    print("CALCULATION COMPLETE")




def reciept():                                               #when called this function
writes and then opens a txt file called reciept.txt
    Reciept = open("reciept.txt", "w+")
    Reciept.write("TRANSACTION RECIEPT\n")                                  #with all
information of the transaction printed on it
    Reciept.write("POUNDS GIVEN          ")
    Reciept.write(str(GBP))
    Reciept.write("\n")
    Reciept.write(Currency.upper())
    Reciept.write(" RECIEVED          ")
    Reciept.write(str(Exchanged))
    Reciept.write("\nTRANSACTION FEE        ")
```

```
    Reciept.write(str(TransactionFee))
    Reciept.write("\n TOTAL COST            ")
    Reciept.write(str(TotalFee))
    Reciept.write("\nAPPLICABLE DISCOUNT      ")
    Reciept.write(str(DiscountedMoney))
    Reciept.write("\nFINAL PAYABLE          ")
    Reciept.write(str(Discount))
    Reciept.close()
    os.startfile("reciept.txt", "open")



print("WELCOME TO THE CASH CONVERTER")
#simple welcome screen with the current date and time
print(datetime.now())


while V == True:
    w = True
    X = True
    Y = True                                                    #these four are looping
variables which allow for while loops to occur when a user enters incorrect data
    Z = True



    while w == True:
        Start = str(input("DO YOU WISH TO CONVERT YOUR CASH?\n"))
        if Start.upper() == "Y" or Start.upper() == "YES":                    #since the
whole programme is in an infinite loop this input / if statment allows for the user
            w = False                                                   #to quit at the start or end
of a transaction
        elif Start.upper() == "N" or Start.upper() == "NO":
            print("OK THANK YOU FOR USING THIS CASH CONVERTER")
```

```
        time.sleep(1)
        print("GOODBYE!")
        time.sleep(3)
        exit()
    else:
        print("ERROR INPUT NOT RECOGNISED. PLEASE TRY AGAIN")


while X == True:
    try:
        GBP = int(input("PLEASE ENTER THE AMOUNT IN POUNDS YOU WISH TO EXCHANGE\n"))
#lets the user input the number of pounds theyre exchanging as an integer
        if GBP > 2500:                                          #this if statments checks
that the user does not exceed the maximum for exchanging
            print("THAT EXCEEDS THE LIMIT FOR CONVERSION (MAX 2500)")
#the result if they do exceed that
        else:                                                   #however if the value is less
than 2500
            X = False                                           #the loop ends
    except(ValueError):                                         #on the condition that
something that isnt an integer is input
        print("ERROR PLEASE ENTER A VALUE IN WHOLE POUNDS")
#the error message displays and the statements loop


while Y == True:
    Currency = str(input("PLEASE ENTER A CURRENCY TO CONVERT TO\nWE CARRY DOLLARS EUROS
REAL YEN AND LIRA \n"))      #allows user to input the currency they wish to convert to
    if Currency.upper() == "USD" or Currency.upper() == "DOLLARS":
#this else if statments compares what the user input to its "database" of known currencies and
        Exchanged = GBP * 1.4                                   #calculates the
exchanged amount
        Y = False
    elif Currency.upper() == "EUR" or Currency.upper() == "EUROS":
        Exchanged = GBP * 1.14
```

```
        Y = False
    elif Currency.upper() == "BRL" or Currency.upper() == "REAL":
        Exchanged = GBP * 4.77
        Y = False
    elif Currency.upper() == "JPY" or Currency.upper() == "YEN":
        Exchanged = GBP * 151.05
        Y = False
    elif Currency.upper() == "TRY" or Currency.upper() == "LIRA":
        Exchanged = GBP * 5.68
        Y = False
    else:
        print("ERROR INPUT NOT RECOGNISED. TRY AGAIN")


  if GBP <= 300:                                                    #this else if statment
calculates the transaction fee by comparing the amount input to various limits,
      TransactionFee = GBP * 0.035                                  #then calculates
the exchange fee
  elif GBP > 300 and GBP <= 750:
      TransactionFee = GBP * 0.03
  elif GBP > 750 and GBP <= 1000:
      TransactionFee = GBP * 0.025
  elif GBP > 1000 and GBP <= 2000:
      TransactionFee = GBP * 0.02
  else:
      TransactionFee = GBP * 0.015


  TotalFee = GBP + TransactionFee                                   #calculates the
total fee the customer will pay before discounts


  while Z == True:                                                  #validation loop
```

```
    Employee = str(input("IS THE CUSTOMER AN EMPLOYEE? (Y/N) \n"))
#employees are eligable for a 5% discount. this input asks weather the customer is
    if Employee.upper() == "Y" or Employee.upper() == "YES":
#Should the customer be an employee then this if statment will create a 5% discount in the total
        DiscountedMoney = TotalFee * 0.05                                   #however the
spec wants it printed seperatly so thats what it does later
        Discount = TotalFee - DiscountedMoney
        Z = False
    elif Employee.upper() == "N" or Employee.upper() == "NO":
        DiscountedMoney = 0
        Discount = TotalFee
        Z = False
    else:
        print("ERROR INPUT NOT RECOGNISED")


  loading()                                                                 #calls the fake loading
screen keeps the user from being bombarded with all the information at once
  time.sleep(1)
  print(f'YOU WILL RECIEVE {Exchanged:.2f} IN', Currency.upper())
#print functions that display the required information in the programme
  time.sleep(1)
  print(f'THE TRANSACTION FEE FOR THIS EXCHANGE IS {TransactionFee:.2f}')
  time.sleep(1)
  print(f'TOTAL FEE COMES TO {TotalFee:.2f}')
  time.sleep(1)
  print(f'YOU ARE APPLICABLE FOR A DISCOUNT OF {DiscountedMoney:.2f}, THIS BRINGS YOUR
TOTAL FEE TO {Discount:.2f}')
  time.sleep(1)
  print("OPENING RECIEPT FILE")
  time.sleep(2)
  reciept()                                                                 #calls the reciept function
to activate
```

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 robust program being created.

Mark in **band 4 (24 marks)**.

```
#This code creates a variable and allows users to input how much money they wish to convert.
Currency_Amount = int(input("please input how much you wish to convert "))
#creates a varialbe for converting currency
Currency_Conversion = input("please indicate how what currency you want to convert to:
A)American Dollars(USD) B)Euros(EUR) C)Brazillian Real(BRL) D)Japenese Yen(JPY) E)Turkish
Lira(TRY)")
#This code converts the Currency_Amount into the selected currency and displays the value.
if Currency_Conversion == "A":
    Currency_Conversion = str(Currency_Amount * 1.4)
    print("Your exchanged amount is: ")
    print(Currency_Conversion)
elif Currency_Conversion == "B":
    Currency_Conversion =str(Currency_Amount * 1.14)
    print("Your exchanged amount is: ")
    print(Currency_Conversion)
elif Currency_Conversion == "C":
    Currency_Conversion = str(Currency_Amount * 4.77)
    print("Your exchanged amount is: ")
    print(Currency_Conversion)
elif Currency_Conversion == "D":
    Currency_Conversion = str(Currency_Amount * 151.05)
    print("Your exchanged amount is: ")
    print(Currency_Conversion)
elif Currency_Conversion == "E":
    Currency_Conversion = str(Currency_Amount * 5.68)
    print("Your exchanged amount is: ")
    print(Currency_Conversion)

#This code calculates the cost for converting into the different currency and outputs the total.
if Currency_Amount < 300:
    Added_Fee = Currency_Amount * 0.035
    Total_Fee = (Currency_Amount + Added_Fee)
```

```
    print("Your total fee is: ")
    print(Total_Fee)
elif 300 <= Currency_Amount > 750:
    Added_Fee = Currency_Amount * 0.03
    Total_Fee = (Currency_Amount + Added_Fee)
    print("Your total fee is: ")
    print(Total_Fee)
elif 750 <= Currency_Amount > 1000:
    Added_Fee = Currency_Amount * 0.025
    Total_Fee = (Currency_Amount + Added_Fee)
    print("Your total fee is: ")
    print(Total_Fee)
elif 1000 <= Currency_Amount > 2000 :
    Added_Fee = Currency_Amount * 0.02
    Total_Fee = (Currency_Amount + Added_Fee)
    print("Your total fee is: ")
    print(Total_Fee)
elif 2000 < Currency_Amount >= 2500:
    Added_Fee = Currency_Amount * 0.015
    Total_Fee = (Currency_Amount + Added_Fee)
    print("Your total fee is: ")
    print(Total_Fee)
#This code varifies if there is a staff discount and takes 5% off of the total cost if applicable.It then
displays the total.
print("are you a staff memeber? Y/N")
Staff_Discount = input()
if Staff_Discount == "Y":
    Reduced_Fee = Total_Fee - (Currency_Amount * 0.05)
    print("Your reduced fee is:")
    print(Reduced_Fee)
if Staff_Discount == "N":
```

```
print("Your total fee is: ")
print(Total_Fee)
```

The learner has produced a program that meets some of the requirements. The program accepts an input for GBP and currency. The fee is calculated incorrectly and there are no calculations to work out the amount of currency received.

No validation has been used. Outputs are accurate and mostly informative. Mostly accurate syntax and indentation used along with some logical structure. Commenting of the code is not very detailed and a third party would have difficulty with it.

Mark in **band 2 (7 marks)**.

Activity 4 – Testing

Test #	Purpose	Data	Expected outcome	actual outcome	Comments
#1	to check that a user cannot enter a value of more than 2500 when entering GBP	entered "90000"	the computer will say that it exceeds the limit and loop back to the input for the GBP	the loop did not take effect	it was a formatting error, forgot to tab in for the section which was looping
#2	To check that the loop does not last infinitely when an appropriate value has been entered	entered "100"	the computer shall move on to the next input of the intended foreign country	As expected	/
#3	To test the input for currency. Firstly to distinguish between a real and made up currency, then to check its loop works appropriately	entered "AAA"	the programme shows an error that it doesn't recognise the input and then loops back to the prior input menu	program continued as though USD had been entered	Missed some formatting for and if / or statement. Needed to set what must = "USD" instead of just or "USD"
#4	to test the employee discount if statements loops correctly	entered "a"	the programme shows an error that it doesn't recognise the input and then loops back to the prior input menu	created an infinite loop	forgot to put the input inside the loop
#5	to test the employee discount if statements loops correctly	entered "y" for discount and 1000 to USD for exchange	the programme subtracts 5% from the total amount and displays it at the end along with the pre-discount total	As expected a pre discounted price of 1025 and post discount 937.75	/
#6	N test for employee discount	entered "n" for the discount and 1000 to USD for exchange	the post and pre discount amount should be the same (1025)	as expected	/
#7	checking that all currencies are calculated properly	entered 1000 into every available foreign currency	1400 in USD, 1140 in EUR, 4770 in BRL, 151050 in JPY and 5680 in TRY	As expected	/
#8	Making sure the receipt works properly	100 usd not an employee	TRANSACTION RECIEPT POUNDS GIVEN 100 USD RECIEVED 140.0	Error function cannot write	Forgot to capitalise on the function causing confusion and hence the error. Easy fix.

			TRANSACTION FEE 3.5000000000000004 TOTAL COST 103.5 APPLICABLE DISCOUNT 0 FINAL PAYABLE 103.5 will be displayed on a text file opened at the end of the program		
#9	Making sure the try except works for the type error in the GBP input	Inputting "aaaaa" into the GBP variable input	Error plese enter an amount in whole pounds will be displayed on screen	As expected	/
#10	Date time check	n/a	After the welcome has displayed the time will be displayed under it	<built-in method now of type object at 0x00000000523262C0> displayed instead	Forgot to add in brackets at the end of datetime.now()
#11	Does the receipt file match the print	100, yen, yes an employee	Receipt and programme will say the same thing however in different formats	The same basic however receipt fills in all significant places.	/
#12	Repetition checking	-	At the endThe programme will repeat without any information from the previous go	As expected	/

The learner has produced 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 (4 marks)**.

Document for Activities 3 and 4

Test Plan (add additional rows as required)

Document for Activities 3 and 4

Test Plan (add additional rows as required)

Program language the product is to be produced in (tick box for language used):

Python ☐ C Family ☐

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
1	To test if a user can enter a number to convert.	123	No error message should be produced	No error is produced.	
2	Test if an error is created when a non-numerical value is entered	One	An error should appear asking for a number.	Produces an error however it breaks the program. Error trapping can be added later to display custom message and not break the program.	
3	To test if the program accepts A, B, C, D, E for selection of conversion currency.	A, B, C, D, E	The program should accept A, B, C, D, and E as valid data for selection of conversion currency.	The program accepts A, B, C, D, E as valid inputs and converts the amount the user wanted to convert and displays it.	
4	To test if the program produces an error message if any other values are entered for the selection of conversion currency. The error message should loop until the correct data is entered.	abcde	The program should produce an error message asking for the user to enter a valid data form of data. The message should loop until a valid entry is made.	The program produces an error but this error breaks the program.	
5	To test if the program produces an error message if a numerical data type is entered. This error message should loop until the correct data type is entered.	356	An error message should be produced asking for the user to input a valid data type. This message should loop until a valid data type is entered.	The program produces an error but doesn't loop. The error breaks the program.	
6	To test if the user is able to enter a numerical value for how much money they wish to convert.	123	The user should be able to enter a numerical value for the amount they wish to convert.	The user can enter a numerical value.	
7	To test if an error message is produced when a non-numerical value is entered.	asd	An error message should appear.	An error if a non-numerical value is entered.	

The learner has identified errors, but comments demonstrate a lack of understanding of the testing process.

Evidence of some errors being resolved is required for higher mark bands.

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

Activity 5

Activity 5: Evaluation

Scenario Requirements

My solution for the scenario covers most of if not all the requirements set out. Firstly it allows the travel agent to enter an amount in pounds that is to be converted. However does not let them exceed 2500 (my 3/pound symbol key isn't working) as that is the maximum that the travel agents can convert in a single transaction. Secondly it allows the user to enter either the short or long form of the currency they wish to convert to (so they can use USD or dollars or EUR or Euros etc.) and the GBP is then converted into that currency. Next based on the pounds input it calculates the transaction fee of between 1.5% and 3.5%. Next it calculates the total cost of the exchange and asks if the customer is an employee, if so it applies a 5% discount to the overall fee. All of this is then printed as an output both in the python shell and as a .txt file. However, the one short coming is that the txt receipt file does not go to 2 decimal places. It does all significant figures. If I had more time I'm sure I could figure this out, however, for now it eludes me. So in general I'd say I have at the very least met the scenario requirements.

Quality and Performance

When it comes to the quality of the code itself I'm sure there are places in which it could have been more efficient, for example, I could've used an array instead of the one variable to shorten down the number of lines of code in the various if statements. The quality of the programme, I believe, is far better than the code and goes beyond the requirements however I would like to have added a web browser function that opened a list of popular tourist attractions in the currencies country. That however can't be done due to inability to access the network. The programme is of a high quality in its robustness as it accepts both short and long inputs for yes no questions and the currency conversion as well as having validation loops at each input and a try, except function when inputting the pounds to be converted. All this create a robust high quality programme that's user friendly and easy to understand. The only problem I would say is that by the second session my 3/pound symbol key stopped working so I couldn't write in the symbol when outputting the receipt.

When it comes to performance I have actually purposefully slowed it down. This is because when it calculates everything and then prints, it all happens very quickly in the blink of an eye, as such I added in a few time.sleep() commands to slow down the outputting to a speed the user can comprehend and not be overwhelmed by. However, if these commands are not present the performance is probably slower than that of someone who did it in less lines of code. But the difference I believe would be negligible.

Coding Conventions

When it comes to the coding conventions I made use of all the basic ones for formatting and have used a wide range of commands in order to create a high quality programme. I have made proper use of indentation and spacing to separate basic functions, I've made use of actual functions as to organise the commands in a way that can best understand, for people who aren't me, I have made use of comments explaining each function in as simple terms as I could think of. I believe my variables were named appropriately and it's fairly easy to interpret what their purpose is For data validation I made breakable while loops that ensured the user entered something applicable before moving on to the next calculation or input. When it came to the GBP integer input I created a try except function to ensure that entering a string wouldn't cause an error and thereby stop the programme. Else if statements were my main way of calculating the exchange and I believe were used appropriately and effectively. Finally as an extra feature I added in a few file handling functions

and imported OS functions in order to write and open a receipt for the user. These all come together to give an easy to understand logically laid out functional programme.

Development Changes

Not much was really changed during the development process. Like I've said before I added in file handling and time displays to add in just some nice aesthetic appeal. Most changes I made were the result of errors in the initial building of the programme. However one change I did make was switching the discount variable, in the pseudo code it was a Boolean but after a while of trying to make that work I couldn't figure it out so I switched it to a string and an if x or y statement so the user can put in a reasonable answer and the programme can apply whatever discount is applicable. The second main difference was the widening of accepted terms for currencies, before I had planned to make the programme only accept the shortened versions such as USD EUR JPY etc. however I had a lot of time left and decided to widen it by firstly figuring out how the .upper() function worked and then adding the long name for the currency into the if statement. Apart from that and adding in some extra features the programme is essentially the same as the pseudocode.

The learner has demonstrated an 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 (8 marks)**.

Evaluation

In this evaluation I shall be talking about 4 main subjects for this assessment. These are:

- How well my solution met the scenario requirements.
- The quality and performance of the program.
- My choice of coding conventions
- Any changes made during the development process.

Firstly, my solution meets the all of the criteria set in the scenario. When the program runs it completes all of the needs and also display: an output of how much the customer receives after conversion, the transaction fee and the total cost. The program is efficient at what it does however it is not robust, if any information is entered that isn't correct it breaks the program. This is due to me not having enough time to produce code within the program to loop code and prevent errors which crash the program. This is primarily due to encountering difficulties whilst I was coding.

Secondly, the program preforms the task that is required of it, however, it is very easy to break due to reasons stated in the previous paragraph. This undoubtedly in my eyes makes this program one that is of a poor standard. If I had more time to work on the program I would be able to do more error checks and implement loops into the code which prevent the program from crashing and/or breaking.

Furthermore, my choices of coding convention make the program easy to understand and allows it to perform the basic tasks that it is required to do. Although, this could be further improved on with the implementations of some arrays as this would allow for less code and more complex error trapping related to inputs which cause the program to crash.

Lastly, I had to make a few changes during the development process. These changes were things like using IF statements a lot more frequently than having an array and using only 1 or 2 IF statements. I also had to rethink my approach to the program, this was due to the way I had originally started coding because it through up a lot of errors and meant that I wasted time by starting again, however, starting again proved more useful because I found an easier method of meeting the criteria, which in turn allowed me to make more progression when coding.

In conclusion the program is poor. I feel with a greater amount of time I could have created a better program that further goes to meet the criteria.

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.

Mark in **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 plans must include normal, abnormal and extreme data. Testing must address errors encountered and how these were overcome.
- Ensure the program uses accurate validation and 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.
- 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.

For more information on Pearson qualifications, please visit
http://qualifications.pearson.com/en/home.html