L3 Lead Examiner Report 1906

Summer 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

Summer 2019
Publications Code 31771H_1906_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	N grade	Level 3		
			Pass	Merit	Distinction
Boundary Mark	0	10	21	34	48

Introduction

This was the third examination season 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 not as good compared to the previous series for this unit. It was evident that some learners were not well prepared for the rigour of this assessment.

The performance on Activity 1 was 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. Learners provided evidence of links between component parts but little evidence of handling errors within the flowcharts.

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 and January 2019.

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

Activity 4 (Coding) was not done to a good standard by a lot of the learners. Some were awarded marks in the top mark band as they produced a working solution along with detailed comments, but most learners produced code that was mark band 2 standard at best. Some learners had not been fully prepared for coding in any programming language and only managed to produce a solution that accepted inputs.

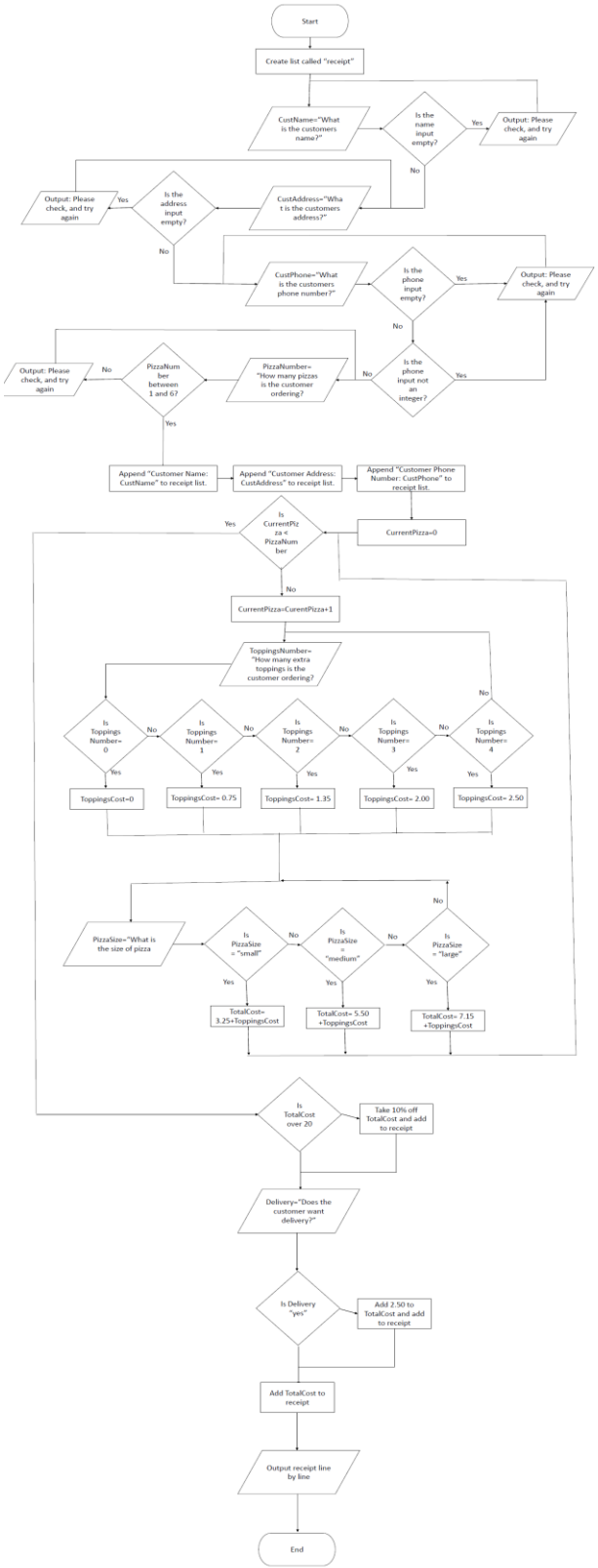
The evaluations (activity 5) were of a good standard and most learners accessed bands two and three. Some learners only produced a review of what they did which resulted in marks from band 1 being awarded.

Individual Questions

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 corresponding mark scheme.

Activity 1

Example response 1

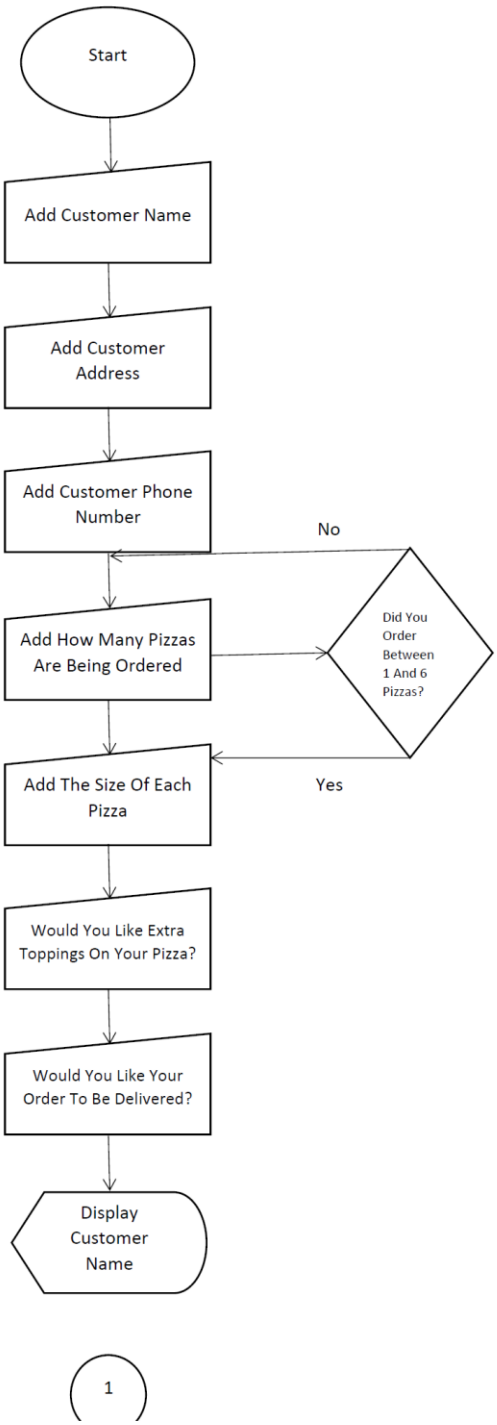


The learner has had a good go at the flow chart for the specified problem.

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 2 (7 marks).**

Example response 2



The learner has addressed all some aspects of the flow chart for the specified problem. British Computer Society (BCS) flowchart symbols have been used but

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

Example response 1

```
Begin
    Total_Cost = 0
    Cust_Name = Input = "Please enter customer name"
    Cust_Address = Input = "Please enter customer address"
    Cust_Phone_Num = int(Input = "Please enter customer phone number")
    Pizza_Quantity = Input = "Please enter number of desired pizzas"
    If Pizza_Quantity >= 1 AND Pizza_Quantity <= 6;
        Then print "Thank you, you have entered an acceptable amount of pizzas"
    Else
        Print "Please enter an acceptable amount of pizzas"

Print "Please enter the desired size of the pizza"
    Pizza_1_Size = int(input = "Press 1 for small, 2 for medium, or 3 for large")
    If Pizza_Quantity > 1
        Pizza_2_Size = int(input = "Select the size of your second pizza: 1 for small, 2 for
        medium, 3 for large")
        If Pizza_Quantity > 2
            Pizza_3_Size = int(input = "Select the size of your third pizza: 1 for small, 2
            for medium, 3 for large")
            If Pizza_Quantity > 3
                Pizza_4_Size = int(input = "Select the size of your fourth pizza: 1 for
                small, 2 for medium, 3 for large")
                If Pizza_Quantity > 4
                    Pizza_5_Size = int(input = "Select the size of your fifth pizza:
                    1 for small, 2 for medium, 3 for large")
                    If Pizza_Quantity > 5
                        Pizza_6_Size = int(input = "Select the size of your
                        final pizza: 1 for small, 2 for medium, 3 for large")
                    Else
                        Print "Thank you, we'll continue with your order"
                Else
                    Print "Thank you, we'll continue with your order"
            Else
```

```
                    Print "Thank you, we'll continue with your order"
            Else
                    Print "Thank you, we'll continue with your order"
        Else
            Print "Thank you, we'll continue with your order"

If Pizza_1_Size = 1
        Then Total_Cost = Total_Cost + 3.25
Elseif Pizza_1_Size = 2
        Then Total_Cost = Total_Cost + 5.50
Else
        Total_Cost = Total_Cost + 7.15

If Pizza_2_Size = 1
        Then Total_Cost = Total_Cost + 3.25
Elseif Pizza_2_Size = 2
        Then Total_Cost = Total_Cost + 5.50
Elseif Pizza_2_Size = 3
        Then Total_Cost = Total_Cost + 7.15
Else
        Total_Cost = Total_Cost

If Pizza_3_Size = 1
        Then Total_Cost = Total_Cost + 3.25
Elseif Pizza_3_Size = 2
        Then Total_Cost = Total_Cost + 5.50
Elseif Pizza_3_Size = 3
        Then Total_Cost = Total_Cost + 7.15
Else
        Total_Cost = Total_Cost
```

```
If Pizza_4_Size = 1
        Then Total_Cost = Total_Cost + 3.25
Elseif Pizza_4_Size = 2
        Then Total_Cost = Total_Cost + 5.50
Elseif Pizza_4_Size = 3
        Then Total_Cost = Total_Cost + 7.15
Else
        Total_Cost = Total_Cost

If Pizza_5_Size = 1
        Then Total_Cost = Total_Cost + 3.25
Elseif Pizza_5_Size = 2
        Then Total_Cost = Total_Cost + 5.50
Elseif Pizza_5_Size = 3
        Then Total_Cost = Total_Cost + 7.15
Else
        Total_Cost = Total_Cost

If Pizza_6_Size = 1
        Then Total_Cost = Total_Cost + 3.25
Elseif Pizza_6_Size = 2
        Then Total_Cost = Total_Cost + 5.50
Elseif Pizza_6_Size = 3
        Then Total_Cost = Total_Cost + 7.15
Else
        Total_Cost = Total_Cost
```

```
Add_Toppings = int(input = "Would you like to add additional toppings to your pizza? Press 1 to add toppings, or press 2 to continue with your order")
If Add_Toppings = 2
    Then print "Thank you, we'll continue with your order"
Elseif Add_Toppings = 1
    Pizza_1_Toppings = int(input = "Please type how many toppings you'd like to add to the first pizza. Enter a number between 0 and 4"
        If Pizza_1_Toppings >= 0 AND <= 4
            Then print "Thank you, we'll add this to your order"
        Else
            Print "Please enter a number between 0 and 4"
    If Pizza_Quantity > 1
        Pizza_2_Toppings = int(input = "Please type how many toppings you'd like to add to the second pizza. Enter a number between 0 and 4"
            If Pizza_2_Toppings >= 0 AND <= 4
                Then print "Thank you, we'll add this to your order"
            Else
                Print "Please enter a number between 0 and 4"
        If Pizza_Quantity > 2
            Pizza_3_Toppings = int(input = "Please type how many toppings you'd like to add to the third pizza. Enter a number between 0 and 4"
                If Pizza_3_Toppings >= 0 AND <= 4
                    Then print "Thank you, we'll add this to your order"
                Else
                    Print "Please enter a number between 0 and 4"
            If Pizza_Quantity > 3
                Pizza_4_Toppings = int(input = "Please type how many toppings you'd like to add to the fourth pizza. Enter a number between 0 and 4"
                    If Pizza_4_Toppings >= 0 AND <= 4
                        Then print "Thank you, we'll add this to your order"
                    Else
                        Print "Please enter a number between 0 and 4"
```

```
If Pizza_Quantity > 4
    Pizza_5_Toppings = int(input = "Please type how many
    toppings you'd like to add to the fifth pizza. Enter a number
    between 0 and 4"
        If Pizza_5_Toppings >= 0 AND <= 4
            Then print "Thank you, we'll add this to your
            order"
        Else
            Print "Please enter a number between 0 and
            4"
    If Pizza_Quantity > 5
        Pizza_6_Toppings = int(input = "Please type how
        many toppings you'd like to add to the final pizza.
        Enter a number between 0 and 4"
            If Pizza_6_Toppings >= 0 AND <= 4
                Then print "Thank you, we'll add
                this to your order"
            Else
                Print "Please enter a number
                between 0 and 4"
    Else
        Print "Thank you, we'll continue with your order"
Else
    Print "Thank you, we'll continue with your order"
Else
    Print "Thank you, we'll continue with your order"
Else
    Print "Thank you, we'll continue with your order"
Else
Print "Please enter either 1 to add toppings, or 2 to continue with your order"
```

```
If Pizza_1_Toppings = 1
        Then Total_Cost = Total_Cost + 0.75
Elseif Pizza_1_Toppings = 2
        Then Total_Cost = Total_Cost + 1.35
Elseif Pizza_1_Toppings = 3
        Then Total_Cost = Total_Cost + 2.00
Elseif Pizza_1_Toppings = 4
        Then Total_Cost = Total_Cost + 2.50
Else
        Total_Cost = Total_Cost

If Pizza_2_Toppings = 1
        Then Total_Cost = Total_Cost + 0.75
Elseif Pizza_2_Toppings = 2
        Then Total_Cost = Total_Cost + 1.35
Elseif Pizza_2_Toppings = 3
        Then Total_Cost = Total_Cost + 2.00
Elseif Pizza_2_Toppings = 4
        Then Total_Cost = Total_Cost + 2.50
Else
        Total_Cost = Total_Cost
```

```
If Pizza_3_Toppings = 1
        Then Total_Cost = Total_Cost + 0.75
Elseif Pizza_3_Toppings = 2
        Then Total_Cost = Total_Cost + 1.35
Elseif Pizza_3_Toppings = 3
        Then Total_Cost = Total_Cost + 2.00
Elseif Pizza_3_Toppings = 4
        Then Total_Cost = Total_Cost + 2.50
Else
        Total_Cost = Total_Cost

If Pizza_4_Toppings = 1
        Then Total_Cost = Total_Cost + 0.75
Elseif Pizza_4_Toppings = 2
        Then Total_Cost = Total_Cost + 1.35
Elseif Pizza_4_Toppings = 3
        Then Total_Cost = Total_Cost + 2.00
Elseif Pizza_4_Toppings = 4
        Then Total_Cost = Total_Cost + 2.50
Else
        Total_Cost = Total_Cost
```

```
If Pizza_5_Toppings = 1
        Then Total_Cost = Total_Cost + 0.75
Elseif Pizza_5_Toppings = 2
        Then Total_Cost = Total_Cost + 1.35
Elseif Pizza_5_Toppings = 3
        Then Total_Cost = Total_Cost + 2.00
Elseif Pizza_5_Toppings = 4
        Then Total_Cost = Total_Cost + 2.50
Else
        Total_Cost = Total_Cost

If Pizza_6_Toppings = 1
        Then Total_Cost = Total_Cost + 0.75
Elseif Pizza_6_Toppings = 2
        Then Total_Cost = Total_Cost + 1.35
Elseif Pizza_6_Toppings = 3
        Then Total_Cost = Total_Cost + 2.00
Elseif Pizza_6_Toppings = 4
        Then Total_Cost = Total_Cost + 2.50
Else
        Total_Cost = Total_Cost

If Total_Cost > 20.00
        Eligible_For_Discount = 1
        Total_Cost = Total_Cost x 0.9
Else
        Eligible_For_Discount = 2
        Total_Cost = Total_Cost
```

```
Request_Delivery = int(input = "Would the customer like to request delivery for an additional cost of £2.50? Press 1 for yes, or press 2 for no")
If Request_Delivery = 1
        Then Total_Cost = Total_Cost + 2.50
Elseif Request_Delivery = 2
        Then Total_Cost = Total_Cost
Else
        Print "Please press either 1 to request delivery at a charge of £2.50, or 2 to not request
        delivery on this order"

Print Cust_Name
Print Cust_Address
Print Cust_Phone_Num
If Pizza_1_Size = 1
        Pizza_1_Price = 3.25
Elseif Pizza_1_Size = 2
        Pizza_1_Price = 5.50
Else
        Pizza_1_Price = 7.15
Print "The price of this size of pizza is £" + Pizza_1_Price

If Pizza_1_Toppings = 1
        Pizza_1_Price = Pizza_1_Price + 0.75
Elseif Pizza_1_Toppings = 2
        Pizza_1_Price = Pizza_1_Price + 1.35
Elseif Pizza_1_Toppings = 3
        Pizza_1_Price = Pizza_1_Price + 2.00
Else
        Pizza_1_Price = Pizza_1_Price + 2.50
Print "The total price for the first pizza including toppings is £" + Pizza_1_Price
```

```
If Pizza_2_Size = 1
        Pizza_2_Price = 3.25
        Print "The price of this size of pizza is £" + Pizza_2_Price
Elseif Pizza_2_Size = 2
        Pizza_2_Price = 5.50
        Print "The price of this size of pizza is £" + Pizza_2_Price
Elseif Pizza_2_Size = 3
        Pizza_2_Price = 7.15
        Print "The price of this size of pizza is £" + Pizza_2_Price
Else
        Pizza_2_Price = 0

If Pizza_2_Toppings = 1
        Pizza_2_Price = Pizza_2_Price + 0.75
Elseif Pizza_2_Toppings = 2
        Pizza_2_Price = Pizza_2_Price + 1.35
Elseif Pizza_2_Toppings = 3
        Pizza_2_Price = Pizza_2_Price + 2.00
Elseif Pizza_2_Toppings = 4
        Pizza_2_Price = Pizza_2_Price + 2.50
Else
        Pizza_2_Price = Pizza_2_Price
If Pizza_2_Price > 0
        Print "The total price for the second pizza including toppings is £" + Pizza_2_Price
```

```
If Pizza_3_Size = 1
        Pizza_3_Price = 3.25
        Print "The price of this size of pizza is £" + Pizza_3_Price
Elseif Pizza_3_Size = 2
        Pizza_3_Price = 5.50
        Print "The price of this size of pizza is £" + Pizza_3_Price
Elseif Pizza_3_Size = 3
        Pizza_3_Price = 7.15
        Print "The price of this size of pizza is £" + Pizza_3_Price
Else
        Pizza_3_Price = 0
If Pizza_3_Toppings = 1
        Pizza_3_Price = Pizza_3_Price + 0.75
Elseif Pizza_3_Toppings = 2
        Pizza_3_Price = Pizza_3_Price + 1.35
Elseif Pizza_3_Toppings = 3
        Pizza_3_Price = Pizza_3_Price + 2.00
Elseif Pizza_3_Toppings = 4
        Pizza_3_Price = Pizza_3_Price + 2.50
Else
        Pizza_3_Price = Pizza_3_Price
If Pizza_3_Price > 0
        Print "The total price for the third pizza including toppings is £" + Pizza_3_Price
```

```
If Pizza_4_Size = 1
        Pizza_4_Price = 3.25
        Print "The price of this size of pizza is £" + Pizza_4_Price
Elseif Pizza_4_Size = 2
        Pizza_4_Price = 5.50
        Print "The price of this size of pizza is £" + Pizza_4_Price
Elseif Pizza_4_Size = 3
        Pizza_4_Price = 7.15
        Print "The price of this size of pizza is £" + Pizza_4_Price
Else
        Pizza_4_Price = 0
If Pizza_4_Toppings = 1
        Pizza_4_Price = Pizza_4_Price + 0.75
Elseif Pizza_4_Toppings = 2
        Pizza_4_Price = Pizza_4_Price + 1.35
Elseif Pizza_4_Toppings = 3
        Pizza_4_Price = Pizza_4_Price + 2.00
Elseif Pizza_4_Toppings = 4
        Pizza_4_Price = Pizza_4_Price + 2.50
Else
        Pizza_4_Price = Pizza_4_Price
If Pizza_4_Price > 0
        Print "The total price for the fourth pizza including toppings is £" + Pizza_4_Price
```

```
If Pizza_5_Size = 1
        Pizza_5_Price = 3.25
        Print "The price of this size of pizza is £" + Pizza_5_Price
Elseif Pizza_5_Size = 2
        Pizza_5_Price = 5.50
        Print "The price of this size of pizza is £" + Pizza_5_Price
Elseif Pizza_5_Size = 3
        Pizza_5_Price = 7.15
        Print "The price of this size of pizza is £" + Pizza_5_Price
Else
        Pizza_5_Price = 0
If Pizza_5_Toppings = 1
        Pizza_5_Price = Pizza_5_Price + 0.75
Elseif Pizza_5_Toppings = 2
        Pizza_5_Price = Pizza_5_Price + 1.35
Elseif Pizza_5_Toppings = 3
        Pizza_5_Price = Pizza_5_Price + 2.00
Elseif Pizza_5_Toppings = 4
        Pizza_5_Price = Pizza_5_Price + 2.50
Else
        Pizza_5_Price = Pizza_5_Price
If Pizza_5_Price > 0
        Print "The total price for the fifth pizza including toppings is £" + Pizza_5_Price
```

```
If Pizza_6_Size = 1
        Pizza_6_Price = 3.25
        Print "The price of this size of pizza is £" + Pizza_6_Price
Elseif Pizza_6_Size = 2
        Pizza_6_Price = 5.50
        Print "The price of this size of pizza is £" + Pizza_6_Price
Elseif Pizza_6_Size = 3
        Pizza_6_Price = 7.15
        Print "The price of this size of pizza is £" + Pizza_6_Price
Else
        Pizza_6_Price = 0
If Pizza_6_Toppings = 1
        Pizza_6_Price = Pizza_6_Price + 0.75
Elseif Pizza_6_Toppings = 2
        Pizza_6_Price = Pizza_6_Price + 1.35
Elseif Pizza_6_Toppings = 3
        Pizza_6_Price = Pizza_6_Price + 2.00
Elseif Pizza_6_Toppings = 4
        Pizza_6_Price = Pizza_6_Price + 2.50
Else
        Pizza_6_Price = Pizza_6_Price
If Pizza_6_Price > 0
        Print "The total price for the final pizza including toppings is £" + Pizza_6_Price


If Eligible_For_Discount = 1
        Print "Customer received a 10% discount on their order"
Else
        Print "Customer received no discount on this order"
```

```
If Request_Delivery = 1
        Print "Delivery charge of "£2.50 was applied"
Else
        Print "No delivery charge applied"

Print "Your total cost for the whole order including any discounts and delivery charges, is £" +
Total_Cost
```

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. Although multiple IF statements have been used, the program will work by will not be as efficient as it could have been if loops had been used.

Mark in **band 3 (8 marks)**.

Example response 2

Pseudocode

Begin

Input Name

Input Address

Input PhoneNumber

Input Amount

Input size of pizza

Input Amount of toppings requested

1 topping equals 75p

2 toppings equal £1.35

3 toppings equal £2.00

4 toppings equal £2.50

5 toppings equal £2.50

6 toppings equal £2.50

Input if user wants order delivered

Print Delivery is £2.50

If size of pizza is small

Price equals £3.25

If size of pizza is medium

Price equals £5.50

If size of pizza is large

Price equals £7.15

Print size of pizza + Amount of toppings

Small pizza + 1 topping = £4.00
Small pizza + 2 toppings = £4.60
Small pizza + 3 toppings = £5.25
Small pizza + 4 toppings = £5.75
Small pizza + 5 toppings = £5.75
Small pizza + 6 toppings = £5.75

Medium pizza + 1 topping = £6.25
Medium pizza + 2 toppings = £6.85
Medium pizza + 3 toppings = £7.50
Medium pizza + 4 toppings = £8.00
Medium pizza + 5 toppings = £8.00
Medium pizza + 6 toppings = £8.00

Large pizza + 1 topping = £7.90
Large pizza + 2 toppings = £8.50
Large pizza + 3 toppings = £9.15
Large pizza + 4 toppings = £9.65
Large pizza + 5 toppings = £9.65
Large pizza + 6 toppings = £9.65

The learner has produced a structure which shows some appropriate and consistent use of hierarchy and indentation, providing some clarity and readable pseudocode. However, the pseudocode will not provide a working solution as it is inefficient and does not produce any outputs.

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

Activity 3 – Test Plan

Example response 1

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
1	to see if when prompted to enter a name, when the right value is entered what it does	"Morgan"	It should proceed through		
2	to see if when prompted to enter a name, when the numerical value is entered what it does	12345	It should through an error and prompt to re-enter again		
3	to see if when prompted to enter a name, when the extreme value is entered what it does	/?'@	It should through an error and prompt to re enter		
4	To see if when prompted to enter an address, when the right value is entered what it does	45 hickenwood avenue,S455rt	It should proceed no errors		
5	To see if when prompted to enter an address, when the extreme value is entered what it does	'{}?>,	It should through an error and prompt to re enter		
6	To see if when prompted to enter an address, when all numeric value is entered what it does	13434555	It should through an error and prompt to re enter		
7	To see if when prompted to enter phone number, when all right value is entered what it does	07864468976	It should proceed through		
8	To see if when prompted to enter a phone number, when all alphabetical value is entered what it does	Maddie	It should through an error and prompt to re enter		

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
9	To see if when prompted to enter a phone number, when an extreme value is entered what it does	@;[].	It should through an error and prompt to re enter		
10	To see when prompted to enter quantity of pizzas, when a normal value is enters what is does	123456	It should proceed through		
11	To see when prompted to enter quantity of pizzas, when an alphabetical value is enters what is does	abcdefgh	It should through an error and prompt to re enter		
12	To see when prompted to enter quantity of pizzas, when a extreme value is enters what is does	789 10	It should through an error and prompt to re enter		
13	To see when prompted to enter size of pizzas, when a normal value is entered what it does	small	It should proceed through		
14	To see when prompted to enter size of pizzas, when a numeric value is entered what it does	123456	It should through an error and prompt to re enter		
15	To see when prompted to enter size of pizzas, when a extreme value is entered what it does	Morgan	It should through an error and prompt to re enter		
16	To see when prompted to enter how many toppings they want, when the normal value is entered what it does	0123456	It should proceed through		
17	To see when prompted to	abdcefgh	It should through an error		

The learner has produced an adequate test plan to confirm a working solution which includes a range of data. Expected results are generic and lack detail but this test plan provides just enough to get into mark band 2.

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

Example response 2

I used C family as my programming language

Test Number	Purpose test	Test Data	Expected Result	Actual Result	Comments
1	To test code is responding to commands and no errors occur	Testing has been done	No errors occur	The code was tested and no errors occurred. Once we realised a small part of the code had been missing originally	When the program had a test run. It ran the whole program without any errors.
2	Ensure no parts of the code is missing	Testing has been done	All parts of the code are visible	Once I realised I missed a small piece of code out of the first test run no parts of the code were missing and	After looking back through the code after an error occurred. I was able to edit the mistake and the program was able to help
3	To show all steps of the code	Each step of the code is present	All steps will be shown within the code	All steps of the code are present.	An error occurred the first time I tested the program due to a small piece of code missing once I changed it the program ran without any faults.
4	Code must fit the criteria's that have been set	Testing has been done on the program to show all criteria's have been met	The code fits the criteria's and can complete the tasks at hand	The program has been specifically built for the tasks he needs it for.	After taking into account what the employer wants from the program. The program that I have built will be able to complete the task.
5	To find any bugs within the code that could affect the program when it is used	Test to find any bugs that could occur within the code	No bugs should be found	No bugs where found when the test run took place.	After evaluating the program when it was complete a test run was conducted and no bugs were found after tasks were completed successfully.

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

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

Activity 4 – Program

Example response 1

```
#Creating the list to be used as a receipt. The "Customer's Recipt" is acting as the title for when it is printed.
receipt=["Customer's Receipt"]
#Initialising variables, and setting default values.
custName=False
custAdd=False
custPhone=False
maxPizza=False
currentPizza=1
totalCost=0


#Defining a function to add the toppings to the reciept. Saves me having to write this whole block
#out three times for each size of pizza.
#This function works by taking the verified inputs from later on, manipulating them based on the cost of
#toppings, and then converting them into a string, to stick onto another string saved as a variable
#which can then easily be appended onto the running total reciept as its own entry
def addToppings(toppings, size, cost):
    #Making this global, so it can edit the totalCost from outside the function.
    global totalCost
    if toppings == 0:
        totalCost=totalCost+cost
        cost=str(cost)
        cost="£"+cost
        output=size+cost
        receipt.append(output)
```

```
    if toppings == 1:                    #Explaining it all here. Applies to the other if
statments also:
        size=size+"+ one extra topping: £"   #Changes the lable for the receipt
        cost=cost+0.75                   #Adds on the cost of the toppings to the total
        totalCost=totalCost+cost         #Adds that to the overall total
        cost=str(cost)                   #Converts it into a string, so that it can be added to
other strings
        output=size+cost                 #Adds both strings together for the formatting of
the reciept
        receipt.append(output)           #Outputs the final formatted text to the reciept
list


    if toppings == 2:
        size=size+"+ two extra toppings: £"
        cost=cost+1.35
        totalCost=totalCost+cost
        cost=str(cost)
        output=size+cost
        receipt.append(output)


    if toppings == 3:
        size=size+"+ three extra toppings: £"
        cost=cost+2
        totalCost=totalCost+cost
        cost=str(cost)
        output=size+cost
        receipt.append(output)


    if toppings == 4:
        size=size+"+ four extra toppings: £"
        cost=cost+2.5
```

```
        totalCost=totalCost+cost
        cost=str(cost)
        output=size+cost
        receipt.append(output)

#While customer name has no value, take the customer name.
while not custName:
    custName=str(input("What is the customer's name? "))

#Same as cust name
while not custAdd:
    custAdd=str(input("What is customer's address? "))

#While the value is not valid, take an integer. If the user doesn't enter an integer, the program throws
#an exception, which is caught, and used to make the user enter an integer again.
#Once they do, the loop breaks
while custPhone==False:
    try:
        custPhone=int(input("What is customer's phone number? (no spaces) "))
    except:
        print("Please enter a valid number")


#Makes the users input have the proper case for names.
custName=custName.title()
#Formats it for the receipt, with a lable.
custName="Customer Name: "+custName

#Same as name
```

```
custAdd=custAdd.title()
#Same as name
custAdd="Customer Address: "+custAdd

#Converts the number into a string, so it can be added to another string
custPhone=str(custPhone)
#Same as name
custPhone="Customer Phone Number: "+custPhone

#Adds all the cutomer information, now correctly formatted, to the reciept list.
receipt.append(custName)
receipt.append(custAdd)
receipt.append(custPhone)

#Takes a number from one to six. If it is given anything else, it throws an exception, causing a loop
loop=True
while loop==True:
    try:
        maxPizza=int(input("How many pizzas is the customer ordering? (Max of 6) "))
        if maxPizza>6 or maxPizza<1:
            raise ValueError
        else:
            loop=False
    except:
        print("Please enter a valid number")

#For every pizza, take the size of the pizza, in a controlled manner as explained above.
for i in range (0,maxPizza):
    print("What size is pizza number",currentPizza,"'small', 'medium' or 'large'")
```

```
loop=True
while loop==True:
    try:
        pizzaSize=str(input())
        if pizzaSize!="small" and pizzaSize!="medium" and pizzaSize!="large":
            raise ValueError
        else:
            loop=False
    except:
        print("Please enter a valid input. 'small', 'medium', 'large'")

#Same again here, but for number of toppings
print("How many extra toppings is the customer having? (0 for none)")
loop=True
while loop==True:
    try:
        toppingsNum=int(input())
        if toppingsNum>4 or toppingsNum<0:
            raise ValueError
        else:
            loop=False
    except:
        print("Please enter a valid number of additional toppings. 1, 2, 3, or 4.")

#This calls the function, and passes it the correct information depending on pizza size.
if pizzaSize=="small":
    tempSize="Small Pizza "
    tempCost=3.25
    addToppings(toppingsNum, tempSize, tempCost)
```

```
    if pizzaSize=="medium":
        tempSize="Medium Pizza "
        tempCost=5.50
        addToppings(toppingsNum, tempSize, tempCost)

    if pizzaSize=="large":
        tempSize="Large Pizza "
        tempCost=7.15
        addToppings(toppingsNum, tempSize, tempCost)

    #Incraments the pizza counter by 1, in a more neat way than the i from the for loop.
    currentPizza+=1

#Checks if the customer wants delivery. Verification method explained above.
loop=True
while loop==True:
    try:
        delivery=str(input("Does the customer need delivery? (yes/no)"))
        if delivery!="yes" and delivery!="no":
            raise ValueError
        else:
            loop=False
    except:
        print("Does the cutomer need delivery? Please enter 'yes', or 'no'")

#Applies a 10% discount on orders over £20. Does this before the delivery charge.
if totalCost>20:
    discount=totalCost*0.1
    totalCost=totalCost-discount
```

```
    #rounds the discount to two decimal places here.
    discount=round(discount, 2)
    #converts it to a string, so that it can be augmented with text.
    discount=str(discount)
    #adds the lable to the number here.
    discount="Discount of 10% applied: -£" + discount
    #adds the number, and the lable, to the receipt.
    receipt.append(discount)

#Very simple, if the user wanted delivery when asked further up the code, it adds it on here.
#It does it down here to it's after the discount calculator.
if delivery=="yes":
    #Adds it directly two the reciept. Not a changing amount, so no variable needed.
    receipt.append("Delivery fee: £2.50")
    #Adds the cost onto the total
    totalCost=totalCost+2.5

#Rounds the total to two places.
totalCost=round(totalCost,2)
#Converts it to a string so it can be added to another string.
totalCost=str(totalCost)
#Adds it to its lable for the receipt.
totalCost="Total cost: £"+totalCost
#Appends it to the very end of the receipt.
receipt.append(totalCost)

#Very simple for loop, prints the reciept line by line. Otherwise it appears with all the [square brackets]
#and commas seperating it, and all next to each other. Was going to use the pprint module, but realised it was even worse.
```

```
for i in receipt:
    print(i)
```

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)**.

Example response 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PizzaCalc
{
   class Program
   {
      static void Main(string[] args)
      {
         int size;
         double sizecost = 0;
         int toppings;
         double toppingsTotal = 0;
         string name;
         string address;
         string postcode;
         string number;
         double totalCost;
         int[] arr = new int[5];


         Console.WriteLine("Please input your name");
         name = Console.ReadLine();
         Console.WriteLine("Please input your address");
         address = Console.ReadLine();
         Console.WriteLine("Please input your postcode");
```

```
postcode = Console.ReadLine();
Console.WriteLine("Please input your number");
number = Console.ReadLine();

    Console.WriteLine("\r\nOrder\r\n" + name +
                "\r\nAdress\r\n" + address +
                "\r\nPostcode\r\n" + postcode +
                "\r\nPhone number\r\n" + number + "\r\n");

Console.WriteLine("How many pizzas?");
string sSize = Console.ReadLine();

int i = Convert.ToInt32(sSize);
int [] pizza = new int [i];

for (int a = 1; a < i + 1; a++)
{
  Console.WriteLine("Please enter your pizza option" + a + "1-6");
   string testArray = Console.ReadLine();

   int g = Convert.ToInt32(testArray);

   int[] tests = new int[g];
   {
      Console.WriteLine("What size pizza?");
      Console.WriteLine("Small?");
      Console.WriteLine("Medium?");
      Console.WriteLine("Large?");

      while (true)
```

```
{
    size = int.Parse(Console.ReadLine());

if (size ==1)
    {
        sizecost = 3.25;
        break;
    }

    if (size == 2)
    {
        sizecost = 5.50;
        break;
    }

    if (size == 3)
    {
        sizecost = 7.15;
        break;
    }
}

Console.WriteLine("Do you want to order any extra toppings?");
Console.WriteLine("1. Yes");
Console.WriteLine("2. No");

    while (true)
{
    Console.WriteLine("How many extra toppings?");
    Console.WriteLine("1 extra toppings?");
```

```
            Console.WriteLine("2 extra toppings?");
            Console.WriteLine("3 extra toppings?");
            Console.WriteLine("4 or more extra toppings?");

            switch (Console.ReadLine())
            {
               case "1":
                  toppingsTotal = 0.75;
                  break;

               case "2":
                  toppingsTotal = 1.35;
                  break;

               case "3":
                  toppingsTotal = 2.00;
                  break;
            }

         }
            if(toppings == 2)
         {
            toppingsTotal = 0;
            break;
         }
         break;
      }
      break;
   }
   totalCost = (sizecost + toppingsTotal);
```

```
            Console.WriteLine(totalCost);
            Console.ReadLine();

        }



    }
}
```

The learner has produced a program that meets some of the requirements. The program accepts an input for name, address and phone number. It also allows the user to enter in number of pizzas but crashes when user enters in the pizza size.

No validation has been used. Outputs are accurate and mostly informative. Some accurate syntax and indentation used along with some logical structure. Commenting of the code has not been done so this can only get a mark in mark band 1.

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

Activity 4 – Testing

Example response

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	If the program can correctly accept the user's name	"Steve" "123" " "	"Steve" should be accepted. "123" should not be accepted " " Should not be accepted	When Steve was entered the code accepted it. When 123 was inputted the code accepted it When nothing was entered the code accepted it	I had forgotten to make sure that the code only accepted letters so I added it `if Name.isalpha():`

2	If the program can correctly accept the user's address	"61 Apple Street"	When "61 Apple Street" is inputted the code should accept it	When "61 Apple Street" was inputted the code accepted it.
				`Please enter your address:`
				`Please enter your address:61 Apple Street`
					Nothing needed to be changed

3	If the program can correctly accept the user's phone number	Any 11-digit long number Not 11 digits Not Digits	When an 11-digit long number is inputted it should be accepted When "1" is entered it shouldn't be accepted When "hello" is inputted it shouldn't be accepted	When "12345678910" was entered the code accepted it When "1" was entered the code didn't accept it When "hello" was inputted the code didn't accept it `Please enter your phone number:`	Nothing needed to be changed

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
4	If the program can correctly accept the user's quantity of pizza	"0" "3" "7" "Yes"	"0" should not work as it is too low "3" should work "7" should not work as it is too high "Yes" should not work as it is a number	When 0 was entered the code didn't accept it as the number was too low When 3 was inputted the code accepted it When 7 was inputted the code didn't work as it was too high When Yes was inputted it didn't work as it isn't a number	Nothing needed to be changed

5	If the program can correctly accept the user's pizza sizes	"small"
		"medium"
		"large"
		"0"
			"small" should be accepted
			"medium" should be accepted
			"large" should be accepted
			"0" should not be accepted
				Small was accepted when entered
				Medium was accepted when entered
				Large was accepted when entered
				The question didn't loop correctly when 0 was entered
					I had forgotten to make sure that the question looped so I added a while statement and made it false when a correct answer was inputted
					`while SizLoop == True:`
					`print("Please enter the`
					`SizeLoop = False`

6	If the program can correctly accept if the user wants to add extra toppings	"yes" "no" "maybe" "4"	"yes" should be accepted "no" should be accepted "maybe" should not be accepted "4" should not be accepted	Yes was accepted No was accepted Maybe was not accepted 4 was not accepted	No changes were made
7	If the program can correctly accept the amount of toppings the user wants	"2" "0" "5"	"2" should be accepted "0" should not be accepted "5" should be accepted	2 was accepted 0 was not accepted 5 was accepted	No changes were made

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments

8	If the program can correctly identify if the total price is over £20	A value worth less than £20
		A value worth more than £20
			Under £20 should not give the user a discount
			Over £20 should give the user a discount
				When the total cost was under 20 it gave them a discount	I used to incorrect symbol so I changed it
					`if TotalCost < 20:`
					`if TotalCost > 20:`

9	If the program can correctly reduce the price by 10% if the cost is over £20	(Any value over £20)	The price should be reduced by 10%	The price was reduced by 10%	No changes needed
10	If the program can correctly accept if the user wants delivery	"Yes" "No"	"Yes" should add the delivery cost to the total price "No" should not add any delivery cost	Yes was accepted and delivery costs were added No was accepted and no costs were added	No changes made
11	If the program can add the correct amount of delivery cost of 2.5	(Any value)	The total cost should be increased by 2.5	The total cost was correctly increased by 2.5	No changes made
12	If the Customer Details can be outputted correctly	"Steve" (An address) (A phone number)	The same information that is inputted at the beginning should be outputted at the end	The same information was correctly printed at the end so the list worked correctly	No changes made

Test Number	Purpose of test	Test Data	Expected Result	Actual Result	Comments
13	If each pizza cost is correctly outputted	Small pizza (costs 3.25)	Each pizza cost should be outputted at the end	The small pizza cost was correctly outputted at the end	No changes
14	If the discount amount is applied	Over £20 worth of pizza	If a discount is applied it should be stated at the end	The discounted amount was displayed at the end	No changes
15	If the program can correctly output the delivery cost	Yes when asked about delivery	If a delivery cost is added it should be stated at the end	The delivery cost was correctly added after the discount	No changes
16	If the final cost is outputted	A full order to see the final cost	At the end the final cost should be correctly outputted	The final cost didn't output correctly	I had forgotten to add the final cost to the list so I added it `Receipt.append("Total Cost: "+TotalCost)`

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)**.

Activity 3/4 – Test Plan (add additional rows as required)

Example response

Document for Activities 3 and 4

Test Plan (add additional rows are required)

Program language the project is to be completed in: Python

Test Number	Purpose of test	Test data	Expected Result	Actual Result	Comments
1	Check input validation in name selection	Leave input blank	Should ask for name again	Asks for name again	Doesn't give a message saying input was incorrect.
2	Checking name validation	"Martha Stewart"	Should proceed	Proceeds to address	
3	Checking validation again for address	Leave input blank	Should prompt for address	Prompts for address	Also doesn't give message.
4	Checking address validation	"17 Highstreet, Brignt, DN14 0RL"	Should proceed	Goes onto number input	
5	Checking validation for phone number	Put "aaa" in box	Should prompt for number again	Asks for number again.	Does give message.
6	Checking validation for phone number	Leave input blank	Should prompt for number again	Asks for number again.	Does give message.
7	Checking validation for phone number	Enter "999999"	Should prompt for number again	Goes on to next input.	No validation against length of input.
8	Checking validation for phone number	Enter "01724280808"	Should proceed	Goes on to next input.	
9	Checking validation for number of pizzas	Leave input blank	Should prompt for number again	Asks again.	Does say "Please enter a valid number"
10	Checking validation for number of pizzas	Put "bbb" in box	Should prompt for number again	Asks again.	Does say "Please enter a valid number"
11	Checking validation for number of pizzas	Enter "999999"	Should prompt for number again	Asks again.	Does say "Please enter a valid number"
12	Checking validation for number of pizzas	Enter "0"	Should prompt again	Asks again.	Does say "Please enter a valid number"
13	Checking validation for number of pizzas	Enter "4"	Should proceed	Goes on to size inputs	
14	Checking size validation	Enter "Small"	Should let you through	Does not let you through. Asks again.	Slight error, only recognises exact inputs.
15	Checking size validation	Enter "small"	Should let you through	Goes onto toppings input.	Prompts again, "Please enter a valid input" then lists valid inputs.
16	Checking size validation	Enter nothing	Should prompt again	Does not let you through. Asks again.	Prompts again, "Please enter a valid input" then lists valid inputs.
17	Checking size validation	Enter "0"	Should prompt again	Does not let you through. Asks again.	Prompts again, "Please enter a valid input" then lists valid inputs.
18	Checking toppings validation	Leave input blank	Should prompt for number again	Prompts again.	"Please enter a valid number of additional toppings. 1, 2, 3, or 4"
19	Checking toppings validation	Enter "999999"	Should prompt for number again	Prompts again.	"Please enter a valid number of additional toppings. 1, 2, 3, or 4"
20	Checking toppings validation	Enter "0"	Should proceed	Goes onto next pizza if there is one.	
21	Checking toppings validation	Enter "4"	Should proceed	Goes onto next pizza if there is one.	
22	Checking toppings validation	Enter -4	Should prompt for number again	Let's you through as though you had no toppings.	Error. Only checked for less than 4. Have corrected.
23	Checking delivery validation	Enter "yes"	Should proceed	Prints receipt.	
24	Checking delivery validation	Enter "no"	Should proceed	Prints receipt.	
25	Checking delivery validation	Enter "aaa"	Should prompt again	Prompts again.	Message "Does the cutomer need delivery? Please enter 'yes', or 'no'"
26	Checking delivery validation	Enter "1"	Should prompt again	Prompts again.	Message "Does the cutomer need

The learner has identified errors, but comments demonstrate a lack of understanding of the testing process and no mention of how these errors can be resolved. 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

Example response 1

Evaluation

How well my solution meets the requirements of the scenario:

My solution fulfils the requirements of the scenario as it successfully can have customer details inputted into the system in an easy manner which is also error checked and validated to make sure the outputs are meaningful. This is generally beneficial to have these error and validation checks throughout the program to ensure data is appropriate and accurate however running these checks before users can continue can also cause a small processing delay (unnoticeable to most) which might leave users frustrated having to wait before getting access to the next part of the program. I have achieved this by using the following line of code in my program "while fullname.isalpha() == False:" which checks if the customer's name consists of alphabetical characters as it should do.

The quantity of pizzas being ordered is set at a maximum of 6 per individual order and a minimum of 1 per order as defined should be done when developing the program in the scenario, this was set as a requirement in the scenario and has definitely been fully fulfilled as demonstrated in the testing phase of my program. I have added a while loop to make sure that any order with more than 6 or less than 1 pizza are automatically dismissed and the user is prompted to enter a new quantity of pizza, this is useful in terms of making sure the system only processes orders that stick to the rules set out in the scenario however this can also be a disadvantage to the system as users are likely to get frustrated by not being able to input exactly what they want. Overall though in conclusion the advantages outweigh the disadvantages and it keeps the system more in line with scenario requirements. It is achieved mainly though the following line of code "while totalpizza < 1 or totalpizza > 6:"

I made sure that all important details inputted (e.g. customer details, order details, et cetera) gave a confirmation of what was inputted before displaying the next question, they were also clearly displayed when the user receives their personalised items bill, I did this to ensure that my program meets the requirements of the scenario effectively. Having the program display inputs as soon as they've been entered helps make sure that data inputted is correct and outputs are meaningful and also that it's easy to use without easily making mistakes without knowing. However having to see so many different confirmation messages at once (and additionally when the bill is displayed to the user) may be confusing for a user making the system less easy to use and efficient. Overall though having the confirmation of selections made implemented into the system make outputted data overall more meaningful and also the system is more efficient as it won't create as many errors despite being a tiny bit more time consuming for users to read confirmation messages before proceeding to the next section of the program.

In conclusion I would say that my solution definitely meets almost every requirement of the scenario making it overall very effective and useful in terms of the scenario. However I could improve the program further by adding more data validation checks that will not accept for example alphabetical information to the pizza toppings amount to reduce errors throughout the program. Even though many of these have already been implemented in other sections of the program e.g. telephone number can only consist of numerical characters.

The quality and performance of my program:

My program after thoroughly being tested is definitely created up to a very high standard. It passed almost all the tests from the test plan with almost identically or similarly to the expected result, no unexpected errors occurred during testing which demonstrates an effective and robust system.

The performance of my program is very good as I've been efficient and used standard programming conventions where appropriate, by doing so this has been extremely useful in improving the overall performance of my program. However there are a few areas throughout my program where I slack with optimisation like for example I store whether delivery has been selected or not twice throughout my program when this could be dramatically reduced to 1 to improve the overall performance of my program. However generally speaking for the majority of my program I have ensured it is optimised and performs well with a few slight exceptions to this.

The choices I made about coding conventions:

I made serval choices about coding conventions during the development of my program. I made sure when naming variables that they'd be relevant and familiar so it wouldn't be difficult for me to remember these throughout writing the code of the program and I also made sure to ensure they were very different compared to other variables and weren't similar to avoid confusion and slowing me down working.

I also used annotations to help me better understand what exactly each sections of my code do and also makes my program easily editable by another developer, having annotations helps me better meet the client requirements of being robust and efficient.

I have used while loops for validation checks throughout my program as these are efficient and robust and also make sure that outputs are meaningful. This definitely meets the scenario criteria by making sure abnormal and extreme data cannot be submitted to the program.

The changes that I made during the development process:

When developing my program I made numerous changes to better meet the requirements of the scenario, I added a data validation check for after customer details have been inputted to the system (e.g. name, address, phone number) to confirm the accuracy of information entered at this initial part of the program to prevent the chance of errors occurring throughout the program and also to prevent people from accidently inputting the wrong details by mistake (user error). This better meets the scenario requirements on the system being efficient and robust and also that outputs are meaningful.

I also decided to add confirmation messages after any value is inputted into the system, for example when the quantity of pizzas being ordered input is entered a message will confirm whether the amount the user inputted is definitely correct highlighting what the user entered. I have done this to reduce the amount of user error that will occur when users use my system to better meet the requirements of the scenario ensuring that the system is easy to use and that outputs are meaningful.

One of the biggest changes that I decided to make during the development process of the program was the implementation of a line of code that would round up prices to make sure they only go to two decimal places, this is so users don't get confused and to make my system not only easy to use but efficient and robust. This makes the program better suited to the scenario requirements as it

makes sure that the program is easy to use for users and also that the outputs are meaningful as well because any currency past two decimal places isn't accurate.

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 (7 marks)**.

Example response 2

Evaluation

My solution meets all requirements of the scenario, it start off by gathering all data about the customer, name, address and phone number it validates the data entered using "Isalpha" and "Isdigit" to ensure that the data entered is correct, if it isn't it displays an error message and tells user to enter the data again.

Then it asks for the amount of each pizza, small, medium and large and if the total value is under 1 or over 6, shows an error message and tells the user to re-enter the data to ensure the final output is correct.

The program then multiplies the cost of the pizzas with the amount ordered.

Then it takes the amount of pizzas through a for loop, asking if the customer has ordered any toppings for each of the pizzas, every time it goes through a pizza it adds cost to the total value.

The produced value then is checked if is eligible for a discount, if it is 10% of the price is subtracted and if it isn't the value stays the same.

Afterwards the user is asked if they are interested in a delivery, if they are £2.50 is added to the total value.

After this all information is displayed on a personalised bill, just like scenario asked. All data is displayed including:

- Name
- Phone Number
- Address
- If discount was applied
- If delivery was selected
- The price of all pizzas
- The total price

Quality of my program meets my expectations, it's very fast, easy to use and is very high quality as it does exactly what it's supposed to, with few **improvements** it could be easily used by a company.

The improvements I would make are quite minor, but could improve my program. I would start by data validating the address to ensure that it's always in the right format, I would also make sure that each of the toppings added to the pizzas are counted as individual variables so it's easier to output in the personalised bill.

During the process I had to make few **changes** instead of adding every possible capitalization of "Yes" I decided to use "Isalpha" algorithm which allowed me to input any version of "Yes" while being recognised by the program.

Additionally, I changed from an If to a for loop for the toppings as it allowed me to add the toppings much easier to the cost of pizzas.

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. This is more of a review rather than an evaluation of the coding concepts.

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 plan 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. Programs must address the majority of the 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.

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

Pearson Education Limited. Registered company number 872828
with its registered office at Edinburgh Gate, Harlow, Essex CM20 2JE