Examiners' Report
Lead Examiner Feedback

June 2023

Pearson BTEC Nationals
In Computing (31771H)
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

June 2023
Publications Code 31771H_2306_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

Introduction

This was the 9th examination series for Level 3 BTEC Computing Unit 4: Software Design and Development Project.

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

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

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

Introduction to the Overall Performance of the Unit

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

The performance on Activity 1 resulted in most learners picking up marks in band 2. Most of the responses used BCS (British Computer Society) symbols, the better responses were able to break down the requirements into relevant parts. Learners provided evidence of links between component parts but evidence of handling errors within the flowcharts was not always present. Many learners tried to fit the entire flowchart on to one page, which tended to make them difficult to follow. Many learners applied validation, mostly in the form of a presence check on the name, address, and phone numbers . A length check was sometimes performed on the phone number, although some learners used incorrect logic at the boundaries resulting in the wrong length being accepted or rejected. Another common error was to reverse the “yes” and “no” labels on a decision box.

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

Activity 3 & 4 (testing) is still an area where learners under perform. Many learners are only demonstrating skills to access mark band 1. It is recommended that centres reinforce what a test plan should consist of and the importance of testing throughout the whole design, which is essential for accessing higher marks. Learners mostly tested the inputs, with some limited data, stating simply that the data would be accepted / rejected. However, many learners did not test the final calculations using specific examples of parcel size and weight. A full test plan should test the category of a parcel by both size and weight and show the final billing that was expected. Costings should be shown in manual calculations, so that the output from the code could be checked against the calculated prices. As a result, errors in the final billing were often missed. In activity 4 some learners included screen shots of actual output, this was very useful in demonstrating that the code had been tested.

Activity 4 (Coding) was well completed by some learners with marks awarded in the top mark band as they produced a working solution along with detailed comments. These learners were able to successfully write code to validate the user inputs, and by testing the code correctly identified errors by entering wrong data types in the program. A common issue with validation was the lack of feedback to user indicating why the input had been rejected. Meaningful error messages would improve the code and the user experience.

The best examples of code had clear logic for the sizing and pricing of parcels, and had robust checks in place to identify errors in data type entry, For example, trapping string data in the dimensions of the parcel. Weaker examples of the code allowed the code to crash at this point.

During the assessment process, the learners code is run when possible, and tested for functionality. It is also read and inspected.

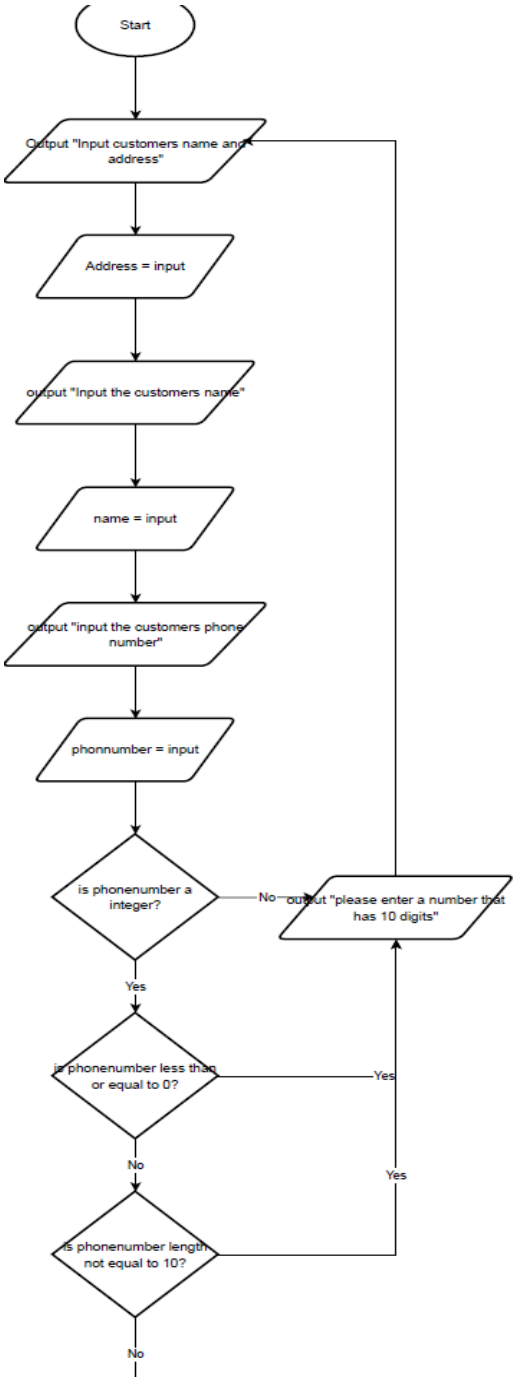
The evaluations (activity 5) were of a good standard and most learner's accessed bands two and three. Some learners only produced a step-by-step account of what they did which resulted in marks from band 1 being awarded.

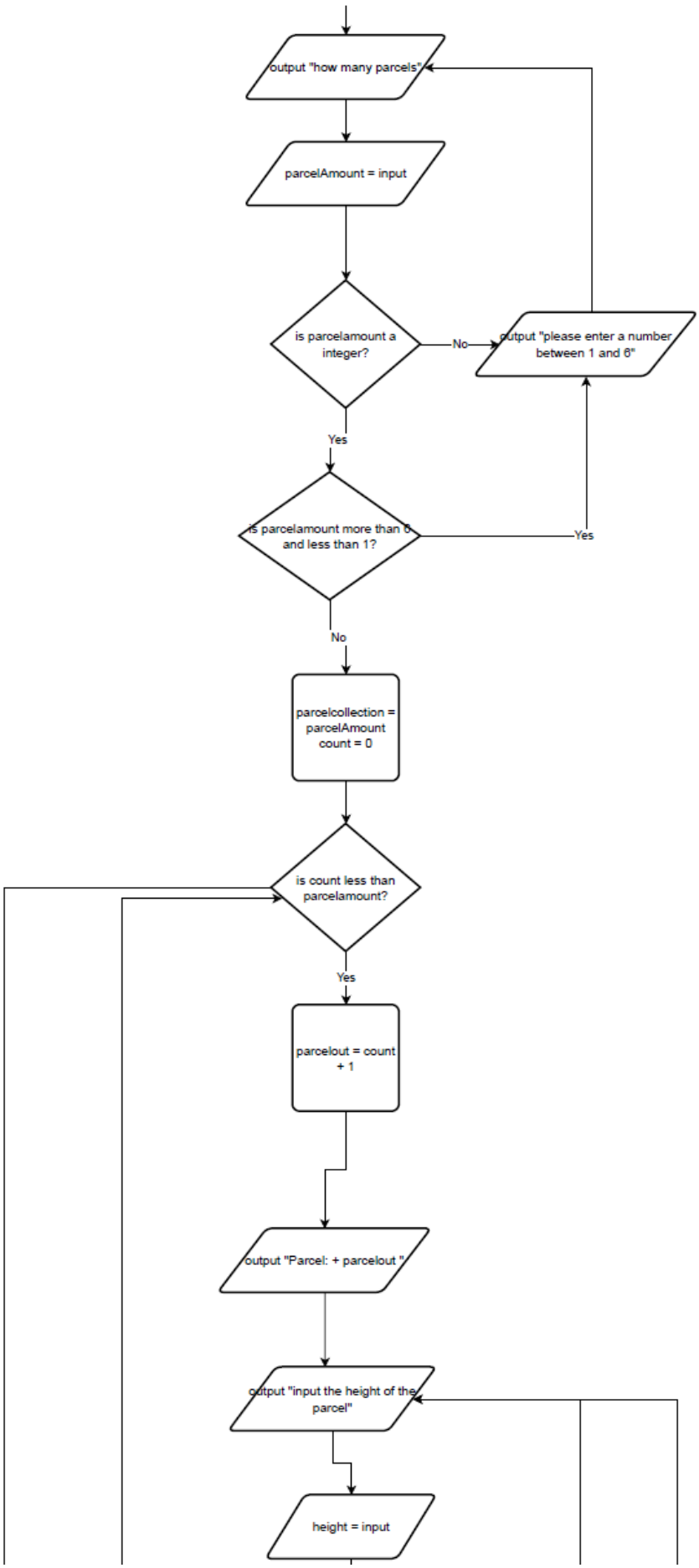
Individual Questions

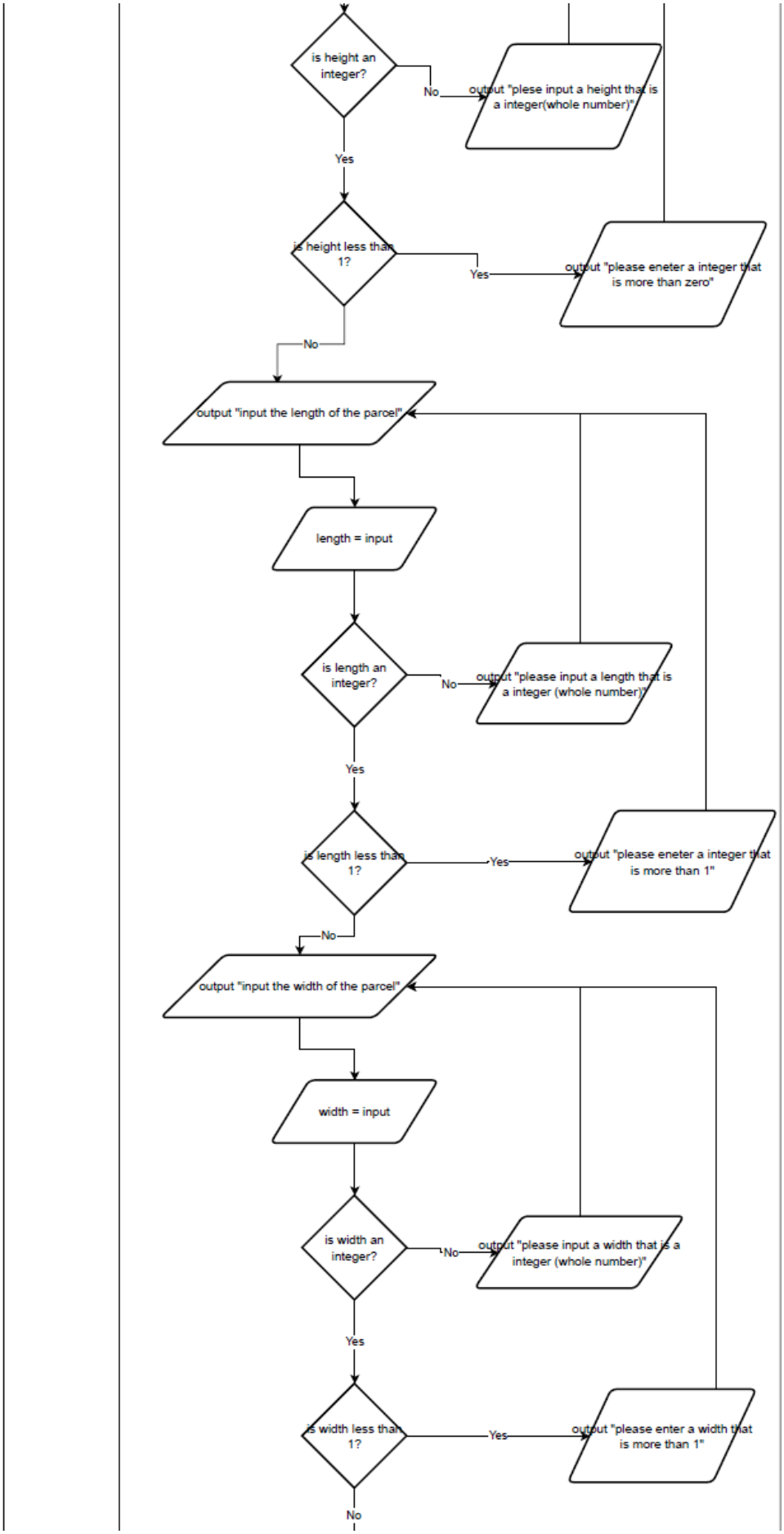
The following section considers each question on the paper, providing examples of learner responses and a brief commentary of why the responses gained the marks they did. This section should be considered with the live external assessment and the corresponding mark scheme.

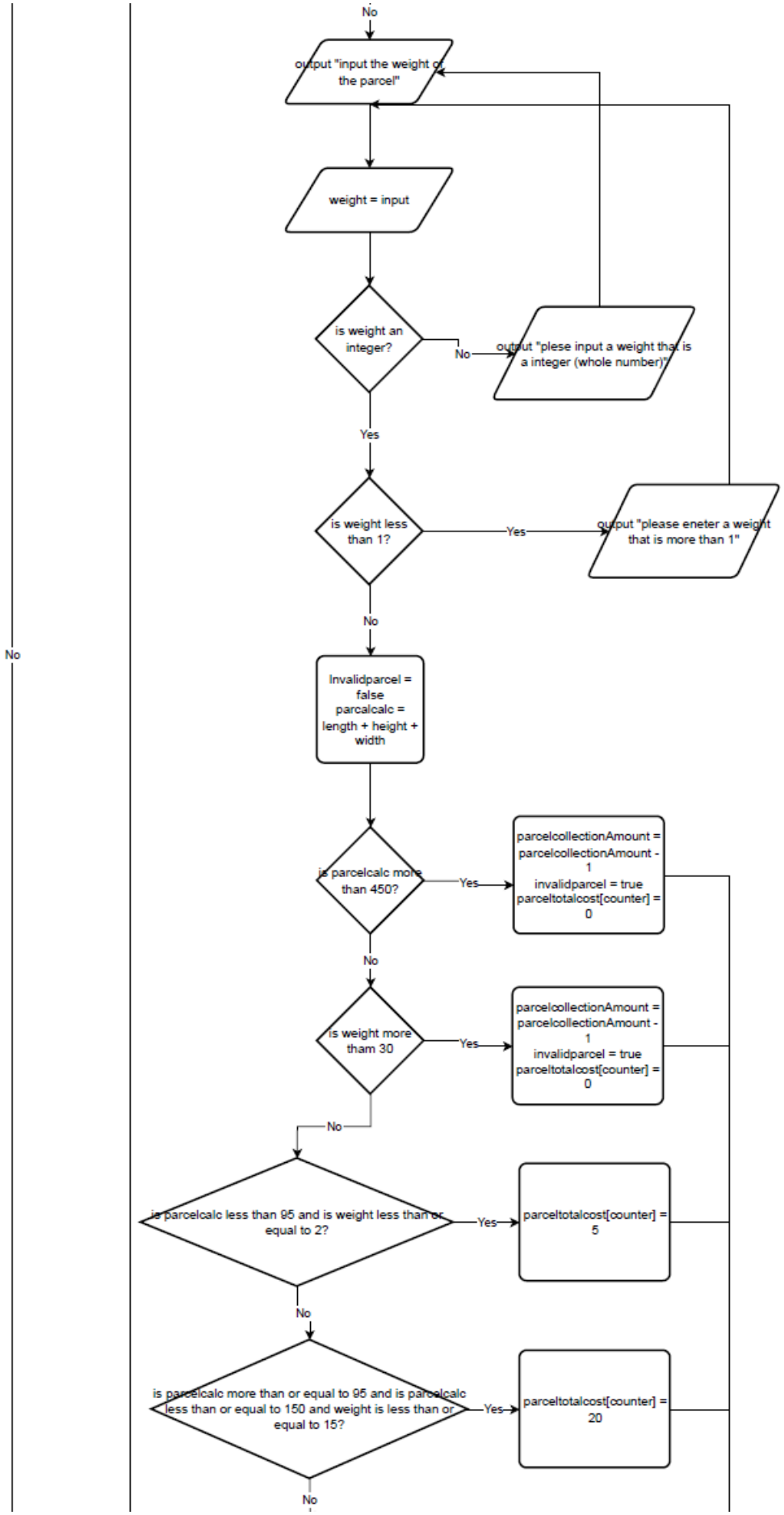
Activity 1

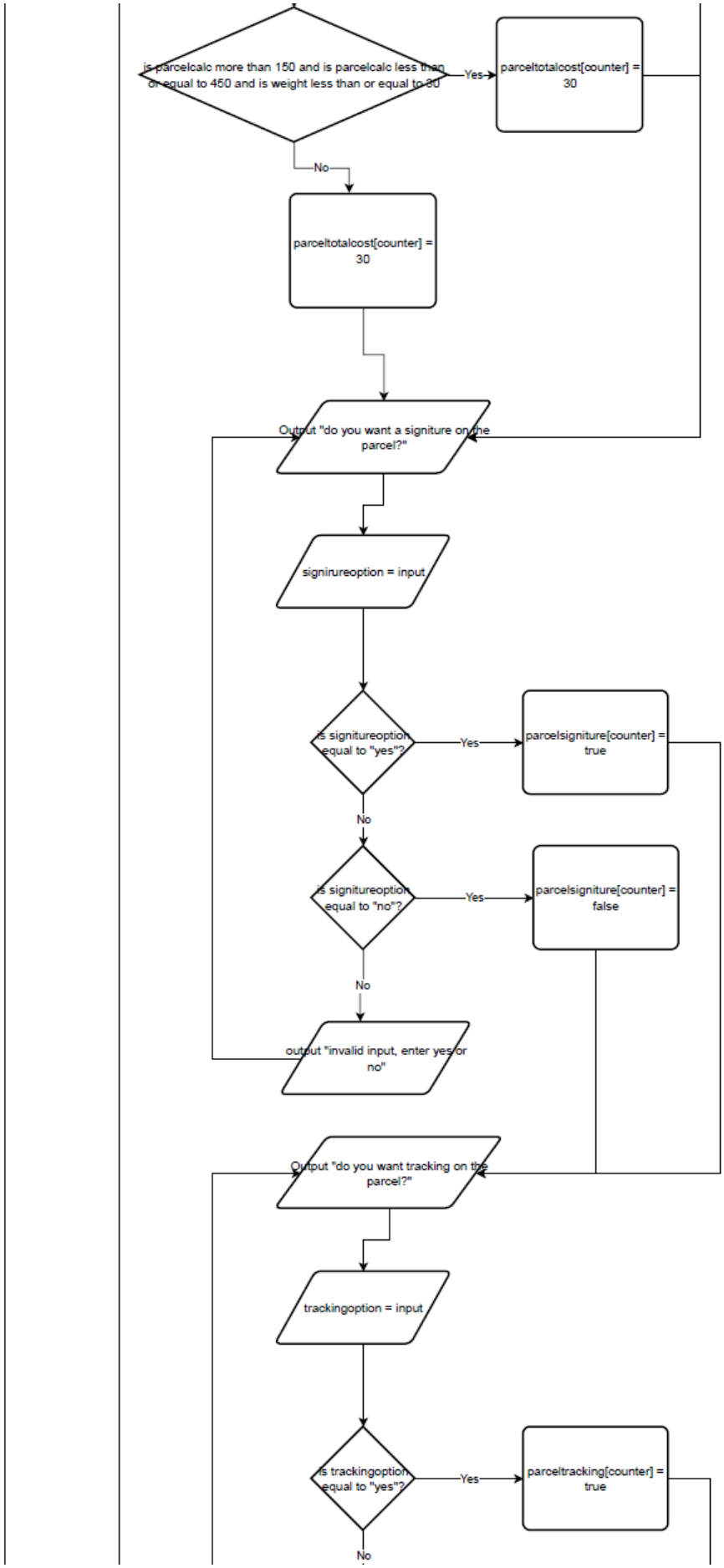
Example 1:

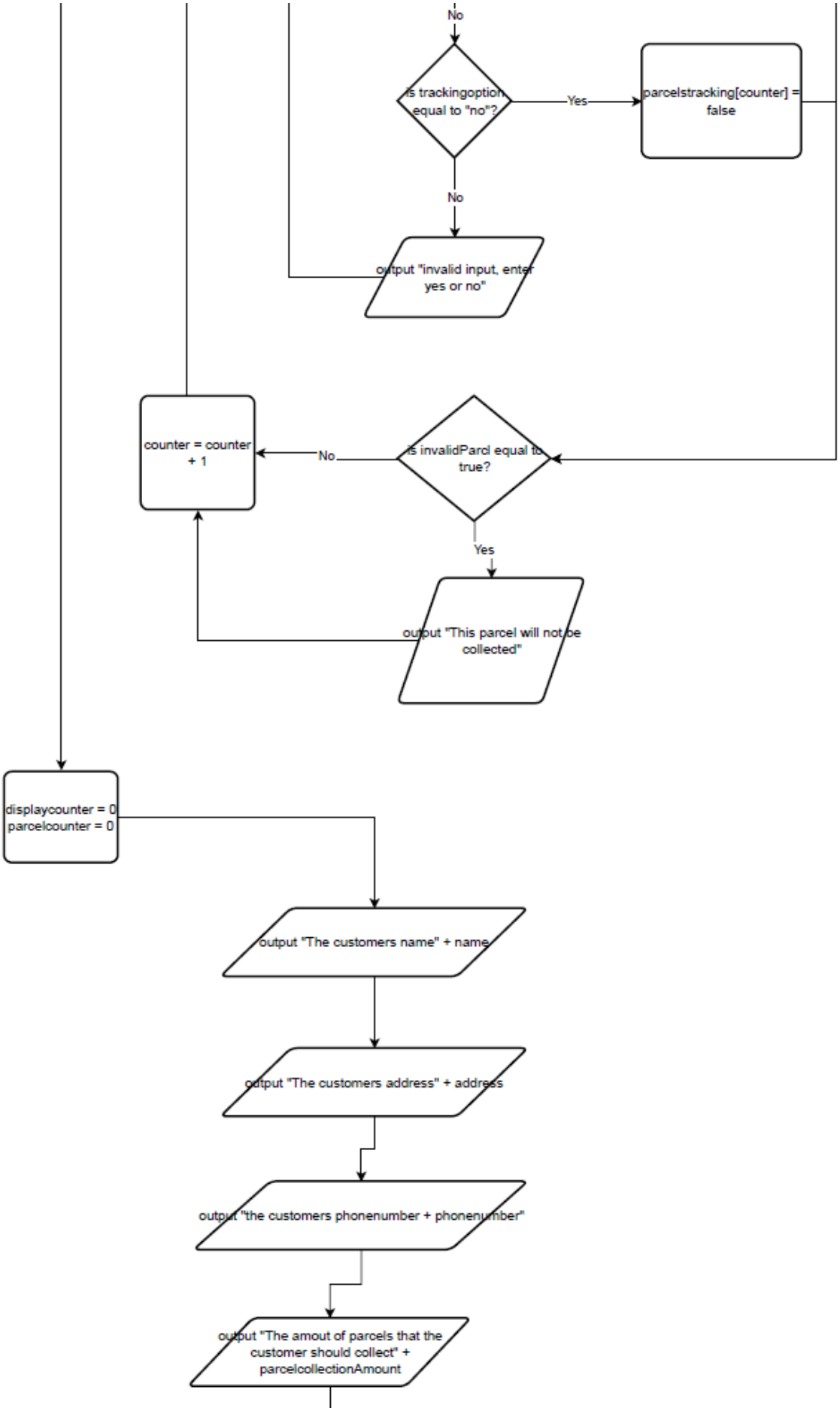


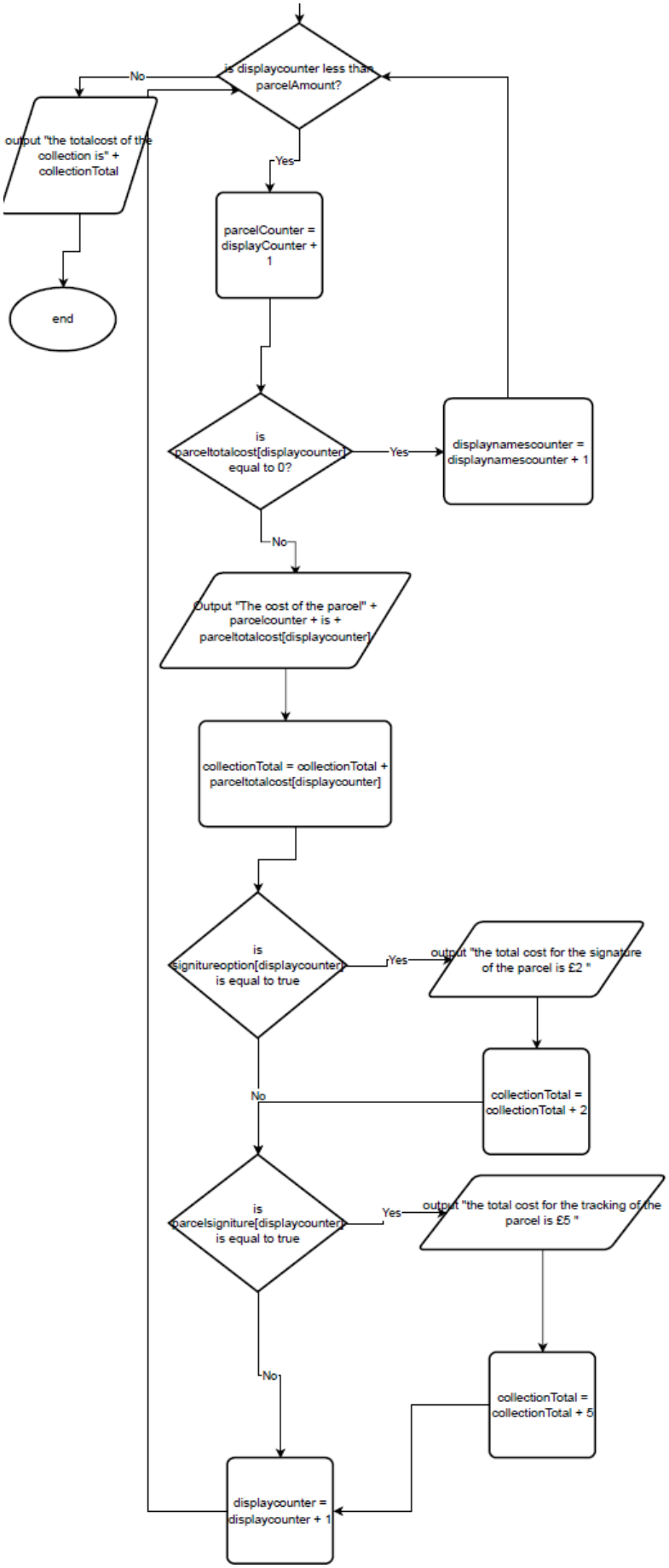

output "how many parcels"
parcelAmount = input
is parcelamount a integer?
No
output "please enter a number between 1 and 6"
Yes
is parcelamount more than 6 and less than 1?
Yes
No
parcelcollection = parcelAmount
count = 0
is count less than parcelamount?
Yes
parcelout = count + 1
output "Parcel: + parcelout "
output "input the height of the parcel"
height = input


is height an integer?
No
output "plese input a height that is a integer(whole number)"
Yes
is height less than 1?
Yes
output "please eneter a integer that is more than zero"
No
output "input the length of the parcel"
length = input
is length an integer?
No
output "please input a length that is a integer (whole number)"
Yes
is length less than 1?
Yes
output "please eneter a integer that is more than 1"
No
output "input the width of the parcel"
width = input
is width an integer?
No
output "please input a width that is a integer (whole number)"
Yes
is width less than 1?
Yes
output "please enter a width that is more than 1"
No


No
output "input the weight of the parcel"
weight = input
is weight an integer?
No
output "plese input a weight that is a integer (whole number)"
Yes
is weight less than 1?
Yes
output "please eneter a weight that is more than 1"
No
No
Invalidparcel = false
parcalcalc = length + height + width
is parcelcalc more than 450?
Yes
parcelcollectionAmount = parcelcollectionAmount - 1
invalidparcel = true
parceltotalcost[counter] = 0
No
is weight more tham 30
Yes
parcelcollectionAmount = parcelcollectionAmount - 1
invalidparcel = true
parceltotalcost[counter] = 0
No
is parcelcalc less than 95 and is weight less than or equal to 2?
Yes
parceltotalcost[counter] = 5
No
is parcelcalc more than or equal to 95 and is parcelcalc less than or equal to 150 and weight is less than or equal to 15?
Yes
parceltotalcost[counter] = 20
No


is parcelcalc more than 150 and is parcelcalc less than or equal to 450 and is weight less than or equal to 30
Yes
parceltotalcost[counter] = 30
No
parceltotalcost[counter] = 30
Output "do you want a signiture on the parcel?"
signirureoption = input
is signitureoption equal to "yes"?
Yes
parcelsigniture[counter] = true
No
is signitureoption equal to "no"?
Yes
parcelsigniture[counter] = false
No
output "invalid input, enter yes or no"
Output "do you want tracking on the parcel?"
trackingoption = input
is trackingoption equal to "yes"?
Yes
parceltracking[counter] = true
No


No
is trackingoption equal to "no"?
Yes
parcelstracking[counter] = false
No
output "invalid input, enter yes or no"
counter = counter + 1
No
is invalidParcl equal to true?
Yes
output "This parcel will not be collected"
displaycounter = 0
parcelcounter = 0
output "The customers name" + name
output "The customers address" + address
output "the customers phonenumber + phonenumber"
output "The amout of parcels that the customer should collect" + parcelcollectionAmount

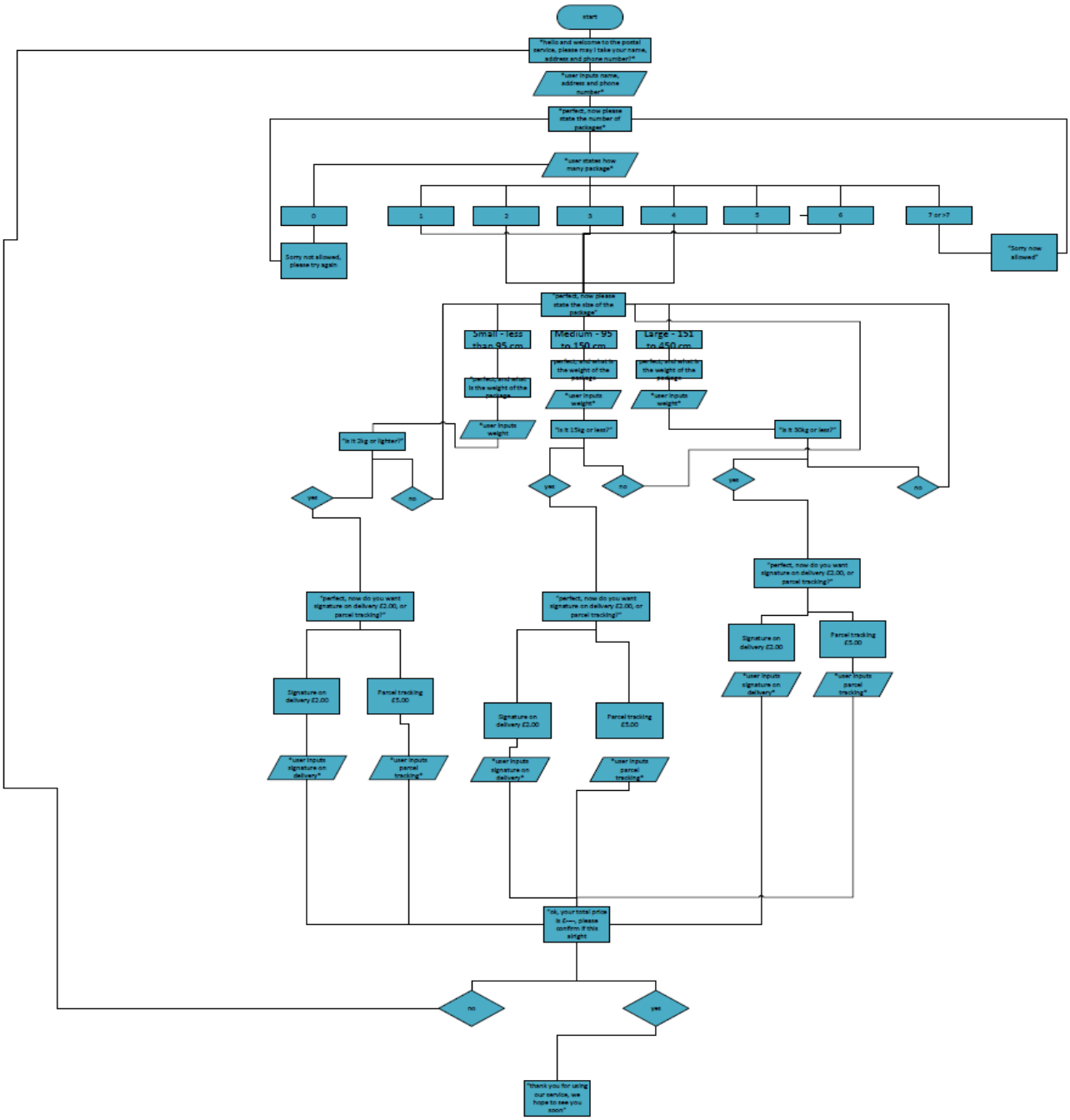

is displaycounter less than parcelAmount?
No
output "the totalcost of the collection is" + collectionTotal
end
Yes
parcelCounter = displayCounter + 1
is parceltotalcost[displaycounter] equal to 0?
Yes
displaynamescounter = displaynamescounter + 1
No
Output "The cost of the parcel" + parcelcounter + is + parceltotalcost[displaycounter]
collectionTotal = collectionTotal + parceltotalcost[displaycounter]
is signitureoption[displaycounter] is equal to true
Yes
output "the total cost for the signature of the parcel is £2 "
No
collectionTotal = collectionTotal + 2
is parcelsigniture[displaycounter] is equal to true
Yes
output "the total cost for the tracking of the parcel is £5 "
No
collectionTotal = collectionTotal + 5
displaycounter = displaycounter + 1

This response shows accurate use of BCS symbols. The logic is correct and the flowchart can be followed, readability could be improved by the use of link symbols between pages.

However, it breaks down the requirements into component parts. There are some inefficient validation processes, but they are mostly correct. The chart shows full coverage of all aspects, the naming conventions used are appropriate and consistent. There are some errors in logic for the parcel sizing. Overall, this response meets the criteria for mark band 3.

Response in **mark Band 3 (8 marks).**

Example 2:



This response shows limited use of BCS symbols and limited coverage of inputs. This is a requirement for this activity (please refer to the first trait in the mark scheme assessment criteria for Activity 1).

The learner is unclear with the validation. However, the flowchart shows that some validation is required. The logic in places is difficult to follow. The chart covers most components of the of solution, but it lacks sufficient detail.

Response in **mark band 2 (4 marks)**.

Activity 2

Example 1:

```
While true
    Output "input the customers name and address"
    Address = input
    Output "input the customers name"
    Name = input
    Try
        Output "input the customers phone number"
        Phonenumber = input

        If phonenumber length is not equal to 10
            Output "please input a phonenumber that has 10 digits"
            Reiterate iteration of loop

        If phonenumber is less than zero
            Output "please input a phonenumber that has 10 digits and is a positive
            integer"
            Reiterate iteration of loop

    Catch Exepction
        Output "please input a phonenumber that is 10 digits long"
        Reiterate iteration of loop

    END WHILE

    While true
        Try
            Output "how many parcels do you want delivered"
            parcelAmount = input

            if parcelAmount is more than 6 and is less than 1
                output "please input a parcel number that is between 1 and 6"
```

```
        else
                END WHILE

Catch Expection
        Output "please input the number of parcels that you want"

parcelcollectionAmount = parcelAmount

counter = 0
while counter is less than parcelAmount
        parcelout = counter + 1
        output "Parcel: " + parcelout
        while true
                try
                        output "input the height of the parcel"
                        height  = input
                        if height is less than 1
                                output "Invalid input, the height needs to be more
                                than zero"
                                Reiterate iteration of loop
                Catch Exception
                        Output "please input a height in centimetre form "
                        Reiterate iteration of loop
                END WHILE

        While true
                Try
                        Output "input the length of the parcel"
                        Length = input
                        If length is less than 1
                                Output "invalid input, the length needs to be more
                                than zero"
```

```
                    Reiterate iteration of loop
            Catch Exception
                    Output "please input the length of the parcel in centimetre
                    form"
                    Reiterate iteration of loop
            END WHILE

While true
        Try
                    Output "input the width of the parcel"
                    Width = input
                    If width is less than 1
                            Output "invalid input, the width needs to be more
                            than zero"
                            Reiterate iteration of loop
        Catch Exception
                    Output "please input the width of the parcel in centimetre
                    form"
                    Reiterate iteration of loop
        END WHILE

While true
        Try
                    Output "please input the weight of the parcel"
                    Weight = input
                    If weight is less than 1
                            Output "invalid input the width needs to be more
                            than zero"
                            Reiterate iteration of loop
        Catch Exception
                    Output "please input the width of the parcel in centimetre
                    form"
        END WHILE
```

```
InvalidParcel = false
Parcelcalc = length + height + width
If parcelcalc is more than 450
        parcelcollectionAmount = parcelcollectionAmount – 1
        parceltotalcost[counter] = 0
        invalidparcel = true
if weight is more than 30
        parcelcollectionAmount = parrcelcollectionAmount – 1
        parceltotalcost[counter] = 0
        invalidparcel = true

else if parcelcalc is less than 95 and weight is less than or equal to 2
        parceltotalcost[counter] = 5
else if parcelcalc is more than or equal to 95 and parcelcalc is less than or
equal to 150 and weight is less than or equal to 15
        parceltotalcost[counter] = 20
else if parcelcalc is more than 150 and parcelcalc is less than or equal to 450
and weight is less than or equal to 30
        parceltotalcost[counter] = 30
else
        parceltotalcost[counter] = 30

while true
        output "do you want a signature on the parcel"
        signitureoption = input
        if signitureoption is equal to yes
                parcelsigniture[counter] = true
                END WHILE
        else if signitureoption is equal to no
                parcelsigniture [counter] = false
                END WHILE
```

```
                else
                        output "Invalid input, please input yes or no"
        while true
                output "do you want tracking on the parcel?"
                trackingoption = input
                if trackingoption is equal to yes
                        parceltracking[counter] = true
                        END WHILE
                else if trackingoption is equal to no
                        parceltracking[counter] = false
                        END WHILE
                else
                        output "please input yes or no"

        if invalidParcel is equal to true
                output "This parcel will not be collected"

        counter = counter + 1

displaycounter = 0
parcelCounter = 0
output "The customers name" + name
output "the customers address: " + address
output "The customers phonenumber" + phonenumber
output "The amount of parcels that the customer should collect" + parcelcollectionAmount

while displaynamescounter is less than parcelAmount
        parcelCounter = displaycounter + 1
        if parceltotalcost[displaycounter] == 0
                displaynamescounter = displaynamescounter + 1
        else
```

```
output "The cost of the parcel " + parcelcounter + "is" +
parceltotalcost[displaycounter]

collectionTotal = collectionTotal + parceltotalcost[displaycounter]


if  signitureoption[displaycounter] is equal to true
        output "the total cost for the signature of the parcel is £2"
        collectionTotal = collectionTotal + 2
if parcelsigniture[displaycounter] is equal to true
        output "The total cost for the tracking of the parcel is £5"
        collectionTotal = collectionTotal + 5
displaycounter = displaycounter + 1


output "The totalcost of the collection is " + collectionTotal
```

The response 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. Appropriate naming conventions have been used, including precise use of logical operations. The validation for the phone number lacks clarity on the looping, but can be interpreted as working. The code covers all aspects of the solution. Calculations are shown in sufficient detail to be followed.

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

Example 2:

BEGIN

Display Customer Name

Input Customer Name

Display Customer Address

Input Customer Address

Display Customer Phone Number

Input Customer Phone Number

While True Do

Display Number of Parcels to collect

Enter Number of Parcels to collect

IF number of parcels to collect is >=1 and <=6

THEN Display number of parcels

ELSEIF Display The minimum number of parcels to collect is 1 and maximum number of parcels to collect is 6

END IF

END ELSEIF

END WHILE

While True Do

Display The height Length and width of the parcel

Input the height length and width of the parcel

IF parcel adds up to <= 95cm

Then it is a small sized parcel

Display the weight of the parcel

Input the weight of the parcel

IF the parcel is <=2kg

ELSE Then the price is £5

ELSEIF the highest price would be applied

IF parcel is >95cm and <150cm

Then it is a medium sized parcel

Display the weight of the parcel

Input the weight of the parcel

IF the parcel is <=15kg

ELSE Then the price is 20kg

ELSEIF the highest price will be applied

IF parcel is >151cm and <450cm

Then it is a large sized parcel

Display the weight of the parcel

Input the weight of the parcel

IF the parcel is <=30kg

ELSE Then the price is £30kg

ELSEIF the highest price will be applied

ELSEIF parcel is >30kg with a size >450cm

ELSE Then parcel will not be collected

END IF

END ELSEIF

Display Does the customer want to add a signature on delivery

Input does the customer want to add a signature on delivery (yes or no)

IF Yes

Then add a cost of £2.00 per parcel for a signature on delivery

ELSEIF no additional cost added

END IF

END ELSEIF

Display Does the customer want to add a parcel tracking

Input Does the customer want to add a parcel tracking (yes or no)

```
IF yes
Then add a cost of £5.00 per parcel for parcel tracking
ELSEIF no additional cost added
END IF
END ELSEIF

Display total cost
End display

Display detailed collection receipt
Display:  Customer Name
Display: Customer address
Display: Customer Phone number
Display: number of parcels to collect
Display: cost of each parcel
Display: cost of signature and tracking for each parcel if required
Display: the total cost of the collection
END DISPLAY
END
```

This response has produced a structure that shows no indents, readability of the hierarchies is therefore limited. The code uses some inconsistent naming conventions. Some validation is shown, including some simple logic. The logic is not always complete or accurate.

Response in **mark band 1 (3 marks)**.

Activity 3

Example 1:

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	Testing to see if the While Loop in the "Customer_Details" Section is functionable	"n" String | Normal	The Loop should continue: Outputting "Enter name:" once the user says the information is incorrect.		
2	Testing to see if the While Loop in the "Customer_Details" Section is functionable	"yes" String | Normal	The Loop should end: Taking the user to next section which is the "Parcels"		
3	Testing to see if the While Loop in the "Customer_Details" Section is functionable and can handle errors	"40" Integer | Abnormal	The Loop should identify that it's the wrong value being entered and therefore should continue back to the start of the program.		
3a	Testing to see if the While Loop in the "Customer_Details" Section is functionable and can handle errors	"40" Integer | Abnormal	The Loop should identify that it's the wrong value being entered and therefore should continue back to the start of the program.		
4	Testing to see if the While Loop in the "Customer_Details" Section is functionable and can handle errors	"*" String | Extreme	The Loop should identify that it's the wrong value being entered and therefore should continue back to the start of the program.		
5	Test the Try and Except Functions as normal without errors	"Bob" String | Normal	The Try and Except function should work in taking you to the next section without having to go through the loop again.		
6	Test the Try and Except Functions to identifying errors	"1.5" Integer | Abnormal	The Try and Except function should work in outputting an error message to the user: "An error has occurred" working with the while loop and taking you back to the beginning.		
7	Test the Try and Except Functions to identifying errors	"Square" for the Phone Number String | Extreme	The Try and Except function should work in outputting an error message to the user: "An error has occurred" working with the while loop and taking you back to the beginning.		
7a	Test the Try and Except Functions to identifying errors	"Square" for the Phone Number String | Extreme	The Try and Except function should work in outputting an error message to the user: "An error has occurred" working with the while loop and taking you back to the beginning.		
8	Test to see if the "Name" variable only accepts strings	"Eryn" String | Normal	The answer should be accepted and move to the first address line.		
9	Test to see if the "Name" variable only accepts strings	"10" Integer | Abnormal	The answer shouldn't be accepted and keep repeating "Enter Name:" until the correct value is entered		

10	Test to see if the "Name" variable only accepts strings	"10/20/23" Date/time | Extreme	The answer shouldn't be accepted and keep repeating "Enter Name:" until the correct value is entered		
11	Test to see if the "address_line1" variable only accepts integers	"2" String | Normal	This answer should be accepted due to it expecting an integer.		
12	Test to see if the "address_line1" variable only accepts integers	"-20" Integer | Abnormal	I do expect this answer to get accepted but I do hope for this to be flagged after the sequence has finished		
13	Test to see if the "address_line1" variable only accepts integers	"House" String | Extreme	This answer should not be accepted as the program is expecting an integer to be inputted.		
14	Test to see if the "address_line2" variable only accepts strings	"Flower Street" String | Normal	The program should accept this answer as it's a string.		
15	Test to see if the "address_line2" variable only accepts strings	"2 Flower Street" String | Abnormal	I do also expect this answer to be accepted but comes down to the user to flag this.		
16	Test to see if the "address_line2" variable only accepts strings	"70" Integer | Extreme	I do not expect this answer to be accepted because it is expecting a string not an integer.		
17	Test to see if the "postcode" variable only accepts strings.	"NE349CF" String | Normal	This string should be accepted by the program and move to the "phone_num" variable		
18	Test to see if the "postcode" variable only accepts strings.	"fork" String | Abnormal	This string will probably get accepted but may require a validation to prove that there are numbers present.		

19	Test to see if the "postcode" variable only accepts strings.	"10" Integer | Extreme	I do not expect this integer to get accepted and should output an error message and take you back to the beginning "Enter name: "		
20	Test to see if the "phone_num" variable only accepts integers	"07395184576" Integer | Normal	The Integer should be accepted and should take you to the next area of the code.		
21	Test to see if the "phone_num" variable only accepts integers	"1" Integer | Abnormal	Due to it being an integer, this would probably be accepted into the system, but down to the user to input the correct information		
22	Test to see if the "phone_num" variable only accepts integers	"Hello" String | Extreme	This will hopefully be flagged and take the user back to the beginning or using the IF statement to confirm their information.		
23	Test to see in the Customer_details if everything has printed out properly	n/a	Everything that is displayed on the screen has been outputted to the user correctly, taking them to the IF statement for their confirmation.		
24	Testing the while loop range of "parcel_amount" is between 1 and 6.	"1" Integer | Normal	The program should continue as normal taking the user to the height, weight, length and width of their parcels.		
25	Testing to see if the range of "parcel_amount" is between 1 and 6.	"6" Integer | Normal	The program should continue as normal taking the user to the height, weight, length and width of their parcels.		

26	Testing to see if the while loop range of "parcel_amount" is between 1 and 6.	"-2" Integer | Abnormal	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.		
27	Testing to see if the while loop range of "parcel_amount" is between 1 and 6.	"7" Integer | Abnormal	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.		
28	Testing to see if the range of "parcel_amount" is between 1 and 6.	"four" String | Extreme	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.		
29	Testing to see if the range of "parcel_amount" is between 1 and 6.	"fifteen" String | Extreme	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.		
30	Test to see if the "Height" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the "Length" Variable		
31	Test to see if the "Height" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.		
32	Test to see if the "Height" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.		
33	Test to see if the "Length" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the "Width" Variable		
34	Test to see if the "Length" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.		
35	Test to see if the "Length" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.		
36	Test to see if the "Width" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the "Weight" Variable		
37	Test to see if the "Width" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.		
38	Test to see if the "Width" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.		
39	Test to see if the "Weight" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the IF Statement		
40	Test to see if the "Weight" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.		
41	Test to see if the "Weight" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.		
42	Check the Size equation to see if it outputs the correct size: Size = height+length+width	"3,3,3" Integers | Normal	The answer should be 9 and be calculated properly, should be placed in the small parcel area		
43	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"3,3,3,2" Integers | Normal	This should be placed in the small parcel area.		

44	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"3,3,3,3" Integers | Abnormal	Should output an error message: "We cannot deliver this"		
45	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"3,3,3,45" Integers | Extreme	Should output an error message: "We cannot deliver this"		
46	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"25,3,121, 3" Integers | Normal	This should be placed in the medium size parcel area.		
47	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"75,25,25, 16" Integers | Abnormal	Should output an error message: "We cannot deliver this"		
48	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"75,25,25, 100" Integers | Extreme	Should output an error message: "We cannot deliver this"		
49	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"150,20,200, 30" Integers | Normal	This should be placed by the IF statement in the large parcel area.		
50	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"150,20,200, 31" Integers | Abnormal	Should output an error message: "We cannot deliver this"		
51	Testing after the Size is calculated, the correct size and weight takes you to the	"150,20,200, -1" Integers | Extreme	Should output an error message: "We cannot deliver this"		

57a	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.		
58	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"70" Integer | Extreme	This answer will not be accepted by the program as it is looking for the "no" or "n" string.		
59	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"yes" String | Normal	This answer should be accepted by the system and apply £5 * number of parcels on the "Total" variable		
60	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"y" String | Abnormal	This answer should be accepted by the system and apply £5 * number of parcels on the "Total" variable		
60a	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"y" String | Abnormal	This answer should be accepted by the system and apply £5 * number of parcels on the "Total" variable		
61	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"70" Integer | Extreme	This answer will not be accepted by the program as it is looking for the "yes" or "y" string. Forcing the program to crash.		
62	Testing the IF Statement of the "parcel_track" variable doesn't apply the "£5" when selecting "no"	"no" String | Normal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.		
63	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next		

	selecting "no"		additional cost of parcel tracking.		
63a	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.		
64	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"70" Integer | Extreme	This answer will not be accepted by the program as it is looking for the "no" or "n" string.		
65	Testing if everything has printed properly for the user, checking if all the calculations are right.	n/a	All of the information printed to the user should be correct data from when they entered it into the system		

The response has produced a test plan to confirm a working solution, which includes a range of data. Expected results are specific and accurate based on identified test data. The examples of inputs are given in the test data and the expected results are described. There is repetitive testing for data input and no real testing of the final solution. This response meets mark band 2.

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

Example 2:

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 [x] C Family [ ]

Test Number	Purpose of test	Test Data	Expected	Actual Result	Comments
1	Test error checking	"abc" - Abnormal	Abnormal - Error message, loop and input again		
2	Parcel size math is correct	Parcel length – 100 - Normal Parcel width – 100 - Normal Parcel height – 100 - Normal	Parcel size = 300		
3	Correct cost added for tracking	Number of parcels – 2 - Normal	Tracking cost = 10		

The plan in this response describes vague testing, and there is minimal test data provided in the table. There is no testing of the logic or validation. The plan is too narrow to confirm a working solution, as only two parts of the calculation is tested.

Response in **mark band 1 (1 mark)**.

Activity 4 Code

Example 1:

```
#Here I initialize two lists to use them later in helping me keep track of the
costs.
parcelCosts = []
parcelDeliverycost = [0, 0, 0, 0, 0, 0]
parcelTrackingcost = [0, 0, 0, 0, 0, 0]


# This while loop collets the customers name
while True:
    customerName = input("Please enter your first name: ")
    customerName = customerName.strip()
    if customerName.isalpha() == True:
        break
    else:
        print("Please enter your first name again!")

#This while loop collects the customers second name - added after testing
while True:
    customerSurname = input("Please enter your surname name: ")
    customerSurname = customerSurname.strip()
    if customerSurname.isalpha() == True:
        break
    else:
        print("Please enter your surname again!")

# This while loop collets the customers phone number
while True:
    customerPhone = input("Please enter your phone number: ")
    customerPhone = customerPhone.strip()
    if len(customerPhone) > 0:
        break
    else:
        print("Please enter your phone number again!")

# This while loop collets the customers address
while True:
    customerAddress = input("Please enter your address: ")
    customerAddress = customerAddress.strip()
```

```
    if len(customerAddress) > 0:
        break
    else:
        print("Please enter your address again!")


# This loop asks the customer how many parcels they want to collect
while True:
    try:
        parcelAmount = int(input("Please enter the number of parcels you wish
to collect: "))

        if parcelAmount < 1 or parcelAmount > 6:
            print("You may only collect between 1 and 6 parcels! Please try
again!")
            continue

        else:
            break

    except:
        print("Please enter a valid number!")


# This loop asks for the parcel sizes and weights and calcuates the cost of
the parcel based on size
for i in range(parcelAmount):
    while True:
        try:
            print(f"These next inputs are for parcel number {i + 1}")
            parcelHeight = int(input("Please enter the height of the parcel in CM:
"))
            parcelWidth = int(input("Please enter the width of the parcel in CM:
"))
            parcelLength = int(input("Please enter the length of the parcel in CM:
"))
            parcelWeight = int(input("Please enter the weight of the parcel in KG:
"))

            parcelSize = parcelHeight + parcelWidth + parcelLength

            if parcelSize > 450:
```

```
            print("Your parcel can not be over 450cm all together!")
            continue

        elif parcelWeight > 30:
            print("Your parcel can not be over 30kg!")
            continue

        elif parcelSize < 95 and parcelWeight <= 2:
            print("Your parcel is size small!")
            parcelCosts.append(5.00)
            break

        elif parcelSize <= 150 and parcelWeight <= 15:
            print("Your parcel is size medium!")
            parcelCosts.append(20.00)
            break

        elif parcelSize <= 450 and parcelWeight <= 30:
            print("Your parcel is size large!")
            parcelCosts.append(30.00)
            break

    except:
        print("Please enter a valid number!")
        continue

# This loop handles the extra charges for signature delivery and tracked
delivery.
for i in range(parcelAmount):
    while True:
        print(f"These next options are for parcel number {i + 1}")
        signatureDelivery = input("Would you like signature on delivery? (Y/N):
")
        parcelTracking = input("Would you like parcel tracking? (Y/N): ")

        answers = ["Y", "N"]
        signatureDelivery = signatureDelivery.upper()
        parcelTracking = parcelTracking.upper()

        if signatureDelivery not in answers or parcelTracking not in answers:
            continue
```

```
        elif signatureDelivery == "Y" and parcelTracking == "Y":
            print("Signature on delivery charge and parcel tracking charge
applied!")
            parcelTrackingcost[i] = 5.00
            parcelDeliverycost[i] = 2.00
            break

        elif signatureDelivery == "Y":
            print("Signature on deliery charge applied!")
            parcelDeliverycost[i] = 2.00
            break

        elif parcelTracking == "Y":
            print("Parcel tracking charge applied!")
            parcelTrackingcost[i] = 5.00
            break

        else:
            break


#This prints out the reciept
print("\n======Customer Receipt======")
print(customerName, customerSurname)
print(customerPhone)
print(customerAddress)

for i in range(parcelAmount):
    print(f"\nThe cost for parcel number {i + 1} is £{parcelCosts[i]}")
    print(f"The extra costs for parcel number {i + 1} is £{parcelDeliverycost[i]}
for signature on delivery and £{parcelTrackingcost[i]} for parcel tracking.")

totalCostone = sum(parcelCosts)
totalCosttwo = sum(parcelTrackingcost)
totalCostthree = sum(parcelDeliverycost)
totalCost = totalCostone + totalCosttwo + totalCostthree
print(f"\nThe total cost for the whole order is £{totalCost}")
```

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

Response in **mark band 4 (24 marks)**.

Example 2:

```
#defines the price of each service and size/weight limits
#defining variables for each size/weight and cost
parcel_sizes = {"small": 95, "medium": 150, "large": 450}
parcel_weights = {"small": 2, "medium": 15, "large": 30}
parcel_cost = {"small": 5.00, "medium": 20.00, "large": 30.00}
signature_cost = 2.0
tracking_cost = 5.0

#menu for the parcel delivery
print(" ")
print("----------------Standard Sizes And Costs Of Parcels ----------------")
print("1. small - less than 95cm(size)| 2kg (max weight) | £5.00 (price)")
print("2. medium - 95 to 150cm (size) | 15kg (max weight) | £20.00 (price)")
print("3. large - 151 to 450cm (size) | 30kg (max weight) | £30.00 (price)")
print("4. signature on delivery - £2.00 per parcel")
print("5. parcel tracking - £5.00 per parcel")
print("6. parcels more than 30kg or with a sizse over 450cm will not be
collected")
print("-----------------------------------------------------------------")
print(" ")


#customers user detail input
#error validation is checked to make sure the user data is correct
while True:
    try:
        print(" ")
        print("CUSTOMER DETAILS -")
        name = str(input("Please enter your name: "))
        address = str(input("Please enter your address: "))
        phone_number = int(input("Please enter your phone number: "))
        break
    except ValueError:
        print("You didn't enter the correct data, Please try again!")


#parcel detail input
#while loop compares the number of parcels thats less than 1 and more
than 6
```

```
print(" ")
print("PARCEL DETAILS - ")
num_parcels = int(input("How many parcels do you have (1-6)? "))
while num_parcels < 1 or num_parcels > 6:
    num_parcels = int(input("Invalid number of parcels, please enter a
number between 1 and 6: "))

parcels = []
total_cost = 0

while True:
    try:
        print(" ")
        for i in range(num_parcels):
            print(f"Enter details for parcel {i+1}:")
            height = float(input("Height (cm): "))
            length = float(input("Length (cm): "))
            width = float(input("Width (cm): "))
            weight = float(input("Weight (kg): "))

    # Check parcel size and weight
    size = "large"
    for key, value in parcel_sizes.items():
        if max(height, length, width) <= value:
            size = key
            break

    if weight > parcel_weights[size]:
        print("Parcels more than 30kg or with a size over 450cm will not be
collected.")
        continue

    # Calculate parcel cost
    cost = parcel_cost[size]
    if input("Add signature on delivery (y/n)? ").lower() == "y":
        cost += signature_cost
    if input("Add parcel tracking (y/n)? ").lower() == "y":
        cost += tracking_cost

    # Add parcel to list and update total cost
    parcels.append((size, cost))
    total_cost += cost
```

```
# Print receipt
print("\n\n==== RECEIPT ====")
print(f"Customer details:\nName: {name}\nAddress: {address}\nPhone
number: {phone_number}")
print(f"\nNumber of parcels: {num_parcels}")
for i, (size, cost) in enumerate(parcels):
    print(f"\nParcel {i+1} details:\nSize: {size}\nCost: £{cost:.2f}")
print(f"\nTotal cost: £{total_cost:.2f}")
```

The response has produced code that uses correct syntax. There are some logic errors in the code that produce infinite loops when run. However there is also some correct logic in other places. The code does not provide a working solution that produces the output required.

Response in **mark band 1 (6 marks).**

Activity 4 Testing

Example 1:

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	Testing to see if the While Loop in the "Customer_Details" Section is functionable	"n" String | Normal	The Loop should continue: Outputting "Enter name:" once the user says the information is incorrect.	The "n" string does work in repeating the while loop to ask the user again to input all the information	
2	Testing to see if the While Loop in the "Customer_Details" Section is functionable	"yes" String | Normal	The Loop should end: Taking the user to next section which is the "Parcels"	The string "yes" does get accepted and asks the user to input the number of parcels they would like.	
3	Testing to see if the While Loop in the "Customer_Details" Section is functionable and can handle errors	"40" Integer | Abnormal	The Loop should identify that it's the wrong value being entered and therefore should continue back to the start of the program.	Sadly, this does get accepted into the system and continue to the next area	
3a	Testing to see if the While Loop in the "Customer_Details" Section is functionable and can handle errors	"40" Integer | Abnormal	The Loop should identify that it's the wrong value being entered and therefore should continue back to the start of the program.	The message still prints out the sentence "Okay!" under the "yes" if statement but it does reset to the beginning of the while loop.	This could also be combined with try and except.
4	Testing to see if the While Loop in the "Customer_Details" Section is functionable and can	"*" String | Extreme	The Loop should identify that it's the wrong value being entered and therefore should continue	This answer does not get accepted by the system, but it does crash the program.	

	handle errors		back to the start of the program.		
5	Test the Try and Except Functions as normal without errors	"Bob" String | Normal	The Try and Except function should work in taking you to the next section without having to go through the loop again.	Due to the program expecting a string, this answer does get accepted and move down to inputting the first address line.	
6	Test the Try and Except Functions to identifying errors	"1.5" Integer | Abnormal	The Try and Except function should work in outputting an error message to the user: "An error has occurred" working with the while loop and taking you back to the beginning.	This answer does get accepted but the user can identify this issue when printing out the information again.	
7	Test the Try and Except Functions to identifying errors	"Square" for the Phone Number String | Extreme	The Try and Except function should work in outputting an error message to the user: "An error has occurred" working with the while loop and taking you back to the beginning.	The program prints out the error message "An error has occurred" and crashes the program"	
7a	Test the Try and Except Functions to identifying errors	"Square" for the Phone Number String | Extreme	The Try and Except function should work in outputting an error message to the user: "An error has occurred" working with the while loop and taking you back to the beginning.	It now detects it as a ValueError and prints out the error message: "An Error has occurred, Please enter the correct information"	I have fixed this issue by using the line "continue" to detect if the user has inputted the correct information.
8	Test to see if the "Name" variable only accepts strings	"Eryn" String | Normal	The answer should be accepted and move to the first address line.	The string is accepted by the system	

9	Test to see if the "Name" variable only accepts strings	"10" Integer | Abnormal	The answer shouldn't be accepted and keep repeating "Enter Name:" until the correct value is entered	The integer was accepted by the system, but the user has the option to repeat the area after the sequence has completed	
10	Test to see if the "Name" variable only accepts strings	"10/20/23" Date/time | Extreme	The answer shouldn't be accepted and keep repeating "Enter Name:" until the correct value is entered	This was also accepted by the system, but the user has the option to flag it at the end of the while loop.	
11	Test to see if the "address_line1" variable only accepts integers	"2" String | Normal	This answer should be accepted due to it expecting an integer.	This answer does accept by the program and continues to address_line2.	
12	Test to see if the "address_line1" variable only accepts integers	"-20" Integer | Abnormal	I do expect this answer to get accepted but I do hope for this to be flagged after the sequence has finished	This answer does get accepted by the system, but the user can double check their information when it gets printed out.	
13	Test to see if the "address_line1" variable only accepts integers	"House" String | Extreme	This answer should not be accepted as the program is expecting an integer to be inputted.	"House" doesn't get accepted by the system and goes back to the start of the WHILE loop.	
14	Test to see if the "address_line2" variable only accepts strings	"Flower Street" String | Normal	The program should accept this answer as it's a string.	As it was expecting a string, this answer was accepted.	
15	Test to see if the "address_line2" variable only accepts strings	"2 Flower Street" String | Abnormal	I do also expect this answer to be accepted but comes down to the user to flag this.	This answer does also get accepted but the user can change their answer before moving onto the amount of parcels.	

16	Test to see if the "address_line2" variable only accepts strings	"70" Integer | Extreme	I do not expect this answer to be accepted because it is expecting a string not an integer.	This answer was accepted but the user can change this before moving on.	
17	Test to see if the "postcode" variable only accepts strings.	"NE349CF" String | Normal	This string should be accepted by the program and move to the "phone_num" variable	Due to this being a string, this answer does get accepted and moves straight on to the phone_num input.	
18	Test to see if the "postcode" variable only accepts strings.	"fork" String | Abnormal	This string will probably get accepted but may require a validation to prove that there are numbers present.	This answer does get accepted but the user has an option to change it before they move on to the parcel amount.	
19	Test to see if the "postcode" variable only accepts strings.	"10" Integer | Extreme	I do not expect this integer to get accepted and should output an error message and take you back to the beginning "Enter name: "	"10" did get allowed by the program but after we print out the information that has been entered, the user has the option to confirm if all of the information is correct.	
20	Test to see if the "phone_num" variable only accepts integers	"07395184576" Integer | Normal	The Integer should be accepted and should take you to the next area of the code.	This did pass through the program due to it being an integer.	
21	Test to see if the "phone_num" variable only accepts integers	"1" Integer | Abnormal	Due to it being an integer, this would probably be accepted into the system, but down to the user to input the correct information	This did get accepted by the program due to it being an integer.	Next time, I would need length validation to ensure the user put the correct among of numbers in to form a phone number,
22	Test to see if the "phone_num" variable only accepts integers	"Hello" String | Extreme	This will hopefully be flagged and take the user back to the beginning or using the IF statement to confirm their information.	"Hello" did cause the error message to be printed and take the user back to the beginning of the WHILE loop	
23	Test to see in the Customer_details if everything has printed out properly	n/a	Everything that is displayed on the screen has been outputted to the user correctly, taking them to the IF statement for their confirmation.	Everything did get printed on the screen properly in a user-friendly format	Checking everything that the user has entered is there on the screen in a user-friendly format
24	Testing the while loop range of "parcel_amount" is between 1 and 6.	"1" Integer | Normal	The program should continue as normal taking the user to the height, weight, length and width of their parcels.	The number one does get accepted by the system and only prints out the next area only once.	
25	Testing to see if the range of "parcel_amount" is between 1 and 6.	"6" Integer | Normal	The program should continue as normal taking the user to the height, weight, length and width of their parcels.	"6" was allowed and prints out the inputs: height, length, width and weight.	
26	Testing to see if the while loop range of "parcel_amount" is between 1 and 6.	"-2" Integer | Abnormal	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.	This answer doesn't get accepted and continues the loop the WHILE loop	
27	Testing to see if the while loop range of "parcel_amount" is between 1 and 6.	"7" Integer | Abnormal	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.	This answer doesn't get accepted and continues the loop the WHILE loop	
28	Testing to see if the range of "parcel_amount" is between 1 and 6.	"four" String | Extreme	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.	This answer does not get accepted by the system and crashes the program.	

29	Testing to see if the range of "parcel_amount" is between 1 and 6.	"fifteen" String | Extreme	I expect the system to output an error message saying it needs a number between 1 and 6. And will continue in the WHILE loop.	This answer does not get accepted by the system and crashes the program.	
30	Test to see if the "Height" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the "Length" Variable	This answer does get accepted and goes to the "height" input.	
31	Test to see if the "Height" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.	"-4" does get accepted by the system and does continue to the "height" input	
32	Test to see if the "Height" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.	The system crashes as it was expecting an integer not a string	
33	Test to see if the "Length" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the "Width" Variable	This answer does get accepted and adds it to the Size variable to calculate the size of the package.	
34	Test to see if the "Length" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.	"-4" does get accepted by the system and does continue to the "Width" input	
35	Test to see if the "Length" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.	The system crashes as it was expecting an integer not a string	
36	Test to see if the "Width" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the "Weight" Variable	This answer does get accepted and adds it to the Size variable to calculate the size of the package.	
37	Test to see if the "Width" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.	"-4" does get accepted by the system and does continue to the "Weight" input	

38	Test to see if the "Width" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.	The system crashes as it was expecting an integer not a string	
39	Test to see if the "Weight" variable only accepts Integers	"6" Integer | Normal	This should be accepted and continue to the IF Statement	This answer does get accepted and adds it to the Size variable to calculate the size of the package.	
40	Test to see if the "Weight" variable only accepts Integers	"-4" Integer | Abnormal	This negative number would properly be accepted by the system but may cause a crash.	"-4" does get accepted by the program and outputs "You have selected a small package"	
41	Test to see if the "Weight" variable only accepts Integers	"boat" String | Extreme	This would cause the system to crash.	The system crashes as it was expecting an integer not a string	
42	Check the Size equation to see if it outputs the correct size: Size = height+length+width	"3,3,3" Integers | Normal	The answer should be 9 and be calculated properly, should be placed in the small parcel area	The calculation was correct and outputs the value "9"	
43	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"3,3,3,2" Integers | Normal	This should be placed in the small parcel area.	After testing, these integers match the requirements of the small parcel size and continues to that part of the IF statement,	
44	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"3,3,3,3" Integers | Abnormal	Should output an error message: "We cannot deliver this"	Outputs the message "We cannot deliver this"	
45	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"3,3,3,45" Integers | Extreme	Should output an error message: "We cannot deliver this"	Outputs the message "We cannot deliver this"	

46	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"25,3,121, 3" Integers | Normal	This should be placed in the medium size parcel area.	This was placed in the medium size parcel area of the IF Statement.	
47	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"75,25,25, 16" Integers | Abnormal	Should output an error message: "We cannot deliver this"	This continues to the medium size parcel area of the IF Statement.	
48	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"75,25,25, 100" Integers | Extreme	Should output an error message: "We cannot deliver this"	Outputs the message "We cannot deliver this"	
49	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"150,20,200, 30" Integers | Normal	This should be placed by the IF statement in the large parcel area.	This was placed in the Large parcel area due to it meeting its requirements.	
50	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price	"150,20,200, 31" Integers | Abnormal	Should output an error message: "We cannot deliver this"	Outputs the message "We cannot deliver this"	
51	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price.	"150,20,200, -1" Integers | Extreme	Should output an error message: "We cannot deliver this"	This continues to the LARGE parcel area.	
52	Testing after the Size is calculated, the correct size and weight takes you to the correct area and applies the right price.	"150,20,200, -1" Integers | Extreme	Should output an error message: "We cannot deliver this"	Now prints out the message "We cannot deliver this"	Just had to add to the elif Weight > 0 and Weight <= 30
53	Testing if the "sign" variable applies the correct additional costs of "£2"	"yes" String | Normal	This answer should be accepted by the system and apply £2 * number of parcels on the "Total" variable	This string does get accepted by the program and applies the £2 per parcel.	Next time, I should've added for how many parcels
54	Testing if the "sign" variable applies the correct additional costs of "£2"	"y" String | Abnormal	This answer should also be accepted by the system and apply £2 * number of parcels on the "Total" variable	This doesn't get accepted by the system as it was looking for the full string "yes"	
54a	Testing if the "sign" variable applies the correct additional costs of "£2"	"y" String | Abnormal	This answer should also be accepted by the system and apply £2 * number of parcels on the "Total" variable	I had to add another variable to add to the additional costs total in order to get it functional.	Similar issues occurred on the sign track, "no" and parcel_track, "yes" and no"
55	Testing if the "sign" variable applies the correct additional costs of "£2"	"70" Integer | Extreme	I do not expect this answer to be accepted by the program and will properly crash it.	This answer wasn't accepted by the program and causes it to crash	
56	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"no" String | Normal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.	This string does get accepted by the program and moves to the parcel_track option	
57	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.	This doesn't get accepted by the system as it was looking for the full string "no"	
57a	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.	I had to add another variable in order to move to the next section.	
58	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"70" Integer | Extreme	This answer will not be accepted by the program as it is looking for the "no" or "n" string.	This answer wasn't accepted by the program and causes it to crash.	

59	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"yes" String | Normal	This answer should be accepted by the system and apply £5 * number of parcels on the "Total" variable	This answer does get accepted by the program and applies the £5 for number of parcels the user has entered.	
60	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"y" String | Abnormal	This answer should be accepted by the system and apply £5 * number of parcels on the "Total" variable	This doesn't get accepted by the system as it was looking for the full string "yes"	
60a	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"y" String | Abnormal	This answer should be accepted by the system and apply £5 * number of parcels on the "Total" variable	I had to add another variable to add to the additional costs total in order to get it functional.	
61	Testing if the "parcel_track" variable applies the correct additional costs of "£5"	"70" Integer | Extreme	This answer will not be accepted by the program as it is looking for the "yes" or "y" string. Forcing the program to crash.	This answer wasn't accepted by the program and causes it to crash.	
62	Testing the IF Statement of the "parcel_track" variable doesn't apply the "£5" when selecting "no"	"no" String | Normal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.	This string does get accepted by the program and moves to the final details, printing all of the information out to the user.	
63	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next additional cost of parcel tracking.	This doesn't get accepted by the system as it was looking for the full string "no"	
63a	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"n" String | Abnormal	The answer should be accepted, and the user should be taken to the next additional cost of parcel	I had to add another variable to add to the additional costs total in order to get it	
			tracking.	functional.	
64	Testing the IF Statement of the "sign" variable doesn't apply the "£2" when selecting "no"	"70" Integer | Extreme	This answer will not be accepted by the program as it is looking for the "no" or "n" string.	This answer wasn't accepted by the program and causes it to crash.	
65	Testing if everything has printed properly for the user, checking if all the calculations are right.	n/a	All of the information printed to the user should be correct data from when they entered it into the system	All the information is printed out correctly in a user-friendly format, easier to read.	

The comments clearly show the process of testing and the output produced, and the testing is adequate to demonstrate the program works. The process shown is largely linear, with some identification of errors and how they were resolved, the comments show only a limited understanding of how errors were found and fixed.

To achieve mark band 2 and above there must be evidence of errors and how they have been solved.

Response in **mark band 2 (4 marks).**

Example 2:

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	Test if the user can enter the right amount	1,2,3,4,5,6	Allows the code to run as normal and move onto collecting data	The user can enter the values between the range and continue to input values	This test is done to ensure the code can run normally without problems
2	Tests the lower and upper bounds of nomParcels	0,7,8,9,10...	Outputs a message saying invalid data then ends the code	The code ends like how expected therefore the user cannot continue to enter any values	This test is done to ensure the user inputs that don't work are denied to stop the code from breaking
3	Test incorrect values (special characters and letters	A,g,.,/,!,@,:	Outputs a message saying invalid data then ends the code	The code crashes but also doesn't let the user enter the values in.	This test is done to ensure the user inputs that don't work are denied to stop the code from breaking
4	Test entering a string into the customerName and address inputs	A,a,haf ahd	Collects the data putting it into the array and then continues running	The user can enter the strings required to continue onto any further inputs to acquire their infomation	This test is done to ensure the code can run normally without problems
5	Test entering numbers and special characters into the string input	1,2,3,4,5,6,7,8,9,0,!,",:,@	Outputs a message saying invalid data then ends the code	The user cannot add any numerical characters into the code however any special characters can be used which needs to be resolved	This test is done to ensure the user inputs that don't work are denied to stop the code from breaking
6	Test whether the array has taken on the values of the entered data.	All the correct data types for the inputs required of the user	The code will take on the values then continue to run like normal	The array has taken upon the values which can be seen with the final product of code since the values get printed out at the end	The code would be changed so they are displayed when entered.
7	Output bill to check if the numbers are accumulating	No test data required other than following the code until the end	The code will count up the numbers and display them in a float format	At first the bill would not take on all the values of the parcels that were entered this would take on only the one value that it would ask for which was easily solved by moving the final print outside of the loop in the correct lines.	This test is done to ensure that the bill is properly added up right without errors
8	Test whether the code can subtract the parcel number from the undelivered	the user to enter the right amount of parcels and follows the code as normal	The code will take away both the values displaying the correct amount of parcels to be delivered.	The code does not run the operation to subtract the undelivered parcels from the ones that will	This test is done to ensure the correct amount of parcels are output to the user too

				be delivered which needs to be fixed.	ensure everything is correct.
9	Test that the loops are working correctly to ensure the user enters all of the packages individually.	All the correct inputs will be needed where asked.	The code asks the user to input all of the parcels separately so that they all have their own dimensions	The code only asked for one of the parcels at first which meant they all had the same dimensions however this was fixed by moving the final output of the code outside of the loops which meant that it would ask for all of them instead of ending.	

Testing shows evidence of a limited testing process, with little identification and resolution of errors. The descriptions used show limited understanding of the testing process.

Response in **mark band 1 (1 mark)**.

Activity 5

Example 1:

"Evaluation

How my solution meets the requirements
I think my solution fully meets the requirements that were given by the scenario. It allows the user to enter all the information they need to then it calculates the cost and then outputs a receipt.
I first start in the program by asking the user to input their name, address and then their phone number making sure to validate the input so that if they make a mistake the program doesn't just crash. If it fails the validation, they are prompted to enter their details again.
I then move onto asking the user about how many parcels they are looking to collect. This is important is the scenario states that they can have a minimum of 1 and a maximum of 6. After entering their amount, I make sure to check it against the requirements and ask the user to re enter if it doesn't.
Thirdly, I ask the user to enter the weights and sizes for each parcel by asking them for the height, length, width and weight. I then use these to check against the boundaries that I have been given in the scenario to work out the cost for each of the parcels. Depending on the size I add the correct cost. I have also made sure to implement the fact that if one of the either size or weight is in a higher boundary than the other that the cost is from the higher one because of what Is stated in the scenario.
I then move onto to checking if the user would like signature on delivery or their parcel tracked. I have made sure to make these checks per parcel as that is what is required. I add the costs depending on the users' answers.
Finally, the last thing that is asked form me in the requirements is to make sure that the user has a receipt containing all the information. I start by printing their personal details so name, phone number and address. I then for each parcel in the order go through it and say what the cost is for it by itself and then let the user know the cost for extra charges if they chose those options. After all of that the final cost of the order all together is displayed.
Quality and performance of my solution
To start with my program, I create two lists that I use to store the values for the cost of each parcel so that I can use them later to print out the bill. I have chosen a list so that I don't have to use lots of different variables. It also makes sense due to the nature of not having a set number of items and this data structure can grow and shrink to fit them.

Secondly, I use a while loop for each of the person detail inputs. As their structure is very similar, I will go through the general similarities and then the difference between the three. I have chosen to use a while loop to start them. This is because I need to have a way to repeat the input as I am going to validate it and make sure that the user has entered good data. I could use a for loop for this job but as I am not sure how many times the user may need to input the data a while loop is a better choice.
For the name input I first use a built-in method that takes away any white space at the start and end of the input. This method is .strip. After this I use an if statement to check if the name contains letters. For this I use another method which is .isalpha and I check if the name with that method is True. If this doesn't evaluate to true, I know that the user has not used letters and I can assume its not a name so I loop back around.
With the number I use the same loop and strip function to remove the white space. I have decided to keep my number as a string instead of changing it to an integer to make sure that the leading 0 that some phone numbers have isn't removed. To validate the data, I check the length of the string after using the strip to do a presence check.
To check the address, I have used the exact same method as the phone number.
I then move onto asking the user the number of parcels that they wish to collect. For this I have used a while loop to make sure I can validate the data but I also used try and except. The reason I used try and except is because I need the input to be an integer. Now if I get the user to enter something and they accidently put a letter the whole program would crash. That's why I needed try except to catch that error and make sure that the program doesn't crash. I then use comparative operators to check if the entered amount is less than or more than the minimum and maximum number of parcels that can be in one order.
To start the process of calculating the price for the parcels I have decided to use a for loop. I have used the for loop because I know how many times, I want this to run. I know I want to run the same number of times as the user wants parcels. So for this I just use I in range of parcelAmount. Inside this for loop I then put a while loop. This may seem inefficient but its because I want to be able to validate the data within and loop back if needs be without messing with the for loop. I will just have to break out of the while loop to continue with the for loop. I start with asking for inputs while inside a try except again to make sure that the program doesn't crash because I want integer inputs and then using these numbers I calculate the total size of the parcel. Now with the parcel size and weight I can use if statements and comparative operators to check against the requirements of the scenario to put the parcel into the correct boundary so the cost is correct.

Now this is where one of the lists from the start comes in because once the boundary has been found I append the cost of the parcel to the list to save it for the receipt. I also use continue and break to control the flow of the loop because if the parcel is over 450cm or 30kg I use continue to make the while loop run again. If the correct boundary has been found for the parcel I use break so that we can move onto the next parcel.
The next section is handling if there is any extra charges per parcel for the signature on delivery or the tracked delivery. I use the same structure as before with the for loop and then the while loop inside of that. I then get the inputs asking the user to use Y/N for each. I have then made a list with Y and N inside it so it can check if the answers are valid. I do this using an if statement and using if answer not in answers. If the code runs down that branch it then continues and does the while loop again. After this I use elif and comparative operators to check what extra charges they want applied. Once the correct if statement has been found I use another list created at the start to store the extra costs.
Finally, I print out the receipt. To start I use \n to make sure that it is on a new line so it is easy to read. I then print the customer details and use another for loop using the amount of parcels to print the cost of the parcel and then the extra costs per parcel. I have used f strings to format the print statements because it is very easy to format with them. I then use the sum method to add up the values in both of the cost lists to give me the total cost. I then finally print the total cost. Coding conventions used
To start off I have been making sure to use PEP8 coding convention to ensure that my code is readable and easy to understand throughout. I will more specifically mention some of the details of that though.
My naming convention for my variables used camelCase. This consists of having two words the first being all lowercase and then the second word having a capital first letter, this is what it looks like myName. This meant that all of my variables were easy to read and understand especially as I was using meaningful names for all of them.
I used comments on my code to make sure that anyone that looks at the code could get an understanding of what the code does just from reading one simple line of text. This is great as a reminder for yourself or for anyone new looking at the code. I have also used very basic terminology so people with low coding knowledge may still be able to understand them.
I have used white space and indentation to ensure that my code looks good and is easy to read. The indents are very important especially in python where a program won't run without the proper indents but the white space is purely to make sure that the code is readable. This means splitting the code up into chunks using white space to separate them.
Changes made during development

During the start of the development, I was stuck on how I wanted to store the costs for each parcel. I thought about having one list to store them all but then later was hit with the realisation that I needed two lists. One for the cost of the parcel dependant on size and weight and one for the extra costs of signature delivery and tracked delivery. This is because in the receipt the charges need to be outputted clearly so the user can see where the cost came from.
I also thought about having the parcel size and weight in the same loop as the parcel extra costs but I thought it would be too much code in one block and would make it so much harder for me to control the flow of the loop efficiently. That would lead to me making more errors and wasting time on a solution that just wasn't a good idea.
During testing I also was made aware of an error I made checking the boundaries for what was an accepted number of parcels to be collected. When 6 was entered (which should be accepted) it asked me to re-enter the amount. I investigated and found there was a problem with my comparative if statement that meant 6 was not accepted. This was my only major error found during testing."

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

Response in **mark band 3** (**8 marks**).

Example 2:

"**Evaluation**
My program solution meets the requirements of the scenario for a parcel collection and delivery service. First of all, it has a user friendly navigation for menu at the start so that its easier to use the program. I believe that my program helps the staff calculate the cost of each parcel delivery and collection overall. It accepts the input for the staff to enter information for customer details such as name, address, phone number. I made sure that I have used the correct data type so that it defines the right type of variable for the customer's user inputs. To make sure the program handles user errors, I applied error handling in the code because this will continue the program even when it faces interruption.
For the parcel details, I have used arithmetic operator to make sure it knows that the users parcel shouldn't be less than 1 or more than 6 in a while loop. When a user enters a number less than 1 or more than 6, it would alert the user with an error message and repeat the number of parcel question once more which means it would execute the code over again until the user inputs the right data. This code gives less line of code which means that it provides high maintainability.
It can also take the height, length, width and weight of each parcel and be able to calculate the total cost of the parcels depending how much the user wants to deliver. It appends each cost and outputs the price at the end of the receipt. For the options, I have asked if the user prefers to add a signature on delivery or to add parcel tracking. However, I could improve on this by allowing the user to enter "n" for no so that it doesn't add extra cost.
My code is maintainable because I have used a while loop which brings less line of code and easier to maintain the program. I also believe that my code is maintainable because it is understandable by using comments on each line of code that states a short explanation of its basic logic of its one block of code. In the future, when another person reads my program and looks at my code, it would increase its readability.
For each line of code, I made sure that it was indented properly, making sure the code was understandable and easier to read. Each code was indented properly on each 'If' statement so that it's easier to read the code. With an extra annotation, I have comments that's brief and specific which explain how the code functions, for instance, a comment on parcel details like "#User input for parcel details".
To make the code more efficient, I have removed less important lines of code, by testing out and developing for loops, while loops, and dictionary. The reason why I have used a for loop was because it would execute a block of code over again which saves time by not writing extra line of code

multiple times. While loop would save more time for users because they wouldn't have to face an error and would navigate back to the same question. To define the cost of every service and size/weight limits, dictionary has benefited the program because it was the best way to put each size and weight together. The only downside is that dictionary can be dynamic which can cause an issue because there is no type checking.
To make sure my code is robust, I ensured that the code I have developed is now fully validated and it can be faced with any unexpected data entry and yet not crash or cause any error. Overall, this would bring less frustration for the users and continue using the program for parcel delivery.
The usability of the program is able to perform correctly depending on the input. The program asks the user to enter between 1 to 6 to allow the user to enter a specific number. Also, for the options of the signature and adding track service, the input asks the user to enter either "y" for yes and "n" for no to make sure they know what to enter without causing any errors in the first place. However, during the testing stage, on each input for adding signature and parcel tracking, I have modified the code by adding .lower() built in method for string handling and be able to return a string where all the characters are in lower case.
Overall, I believe that my program is reliable because I have achieved this by testing out the parcel delivery program to make sure it performs as expected as users' requirements. At the end of the program, the user will understand the outcome of total cost by reading the receipt that I have built that would output their details, number of parcels, size and cost of their parcel and the total cost.
During the development process, I have followed the software development life cycle (SDLC) from start to end to follow the right structure. The six steps that I have followed were conception, analysis, design, implementation, testing and evaluation."

The response shows superficial understanding of relevant technical concepts. The evaluation is mostly descriptive of the process carried out.

There are limited judgements about the quality and performance of the program, which keeps this evaluation response in mark band 1.

mark band 1 (3 marks).

Summary

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

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