Pseudocode

**Task 1**

**Import packages**

IMPORT CSV

**Classes**

Class Store:

FUNCTION INITIALISE Attributes of Store Object

(self, tag, stockitem, numberofweeks):

self.Tag = tag  
self.StockItem = stockitem  
self.Itemsbought = []  
self.alternatives = []

self.dayVisited = 0

for i in range(numberofweeks):  
 self.Itemsbought.append ([])  
 self.alternatives.append ([])

availableitemsinstock(self, item)

BOOLEAN used to check IF the item is available in stock in the store

Class Item

FUNCTION INITIALISE Attributes of ITEM Object

\_\_init\_\_(self, nameofitem, price):

self.NameofItem = nameofItem  
self.Price = price

Class House

FUNCTION INITIALISE Attributes of House Object

\_\_init\_\_(self, housenumber, itemsrequired, numberofweeks):

self.HouseNumber = housenumber  
self.ItemsRequired = itemsrequired  
self.DayofDelivery = []  
for i in range(NumberOfweeks):  
 self.DayofDelivery.append([])

**Global Variables**

CREATE empty list called Stores – an empty list that contains Store objects   
CREATE empty list called Houses - an empty list that contains House objects  
CREATE empty list called Items - an empty list that contains Item objects  
  
Number\_of\_weeks = 0 – An integer to store the number of weeks   
Number\_of\_Houses = 0 – An integer to store the number of weeks

**Helper Functions**

readcsv(filepath):

CREATE an empty list called list\_of\_data

OPEN the csv file located at filepath

READ csv file  
LOOP through each for row in csvreader:  
APPEND the row to a list

RETURN list\_of\_data

daynames():

this is a list and will show the different days of the week

Days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

RETURN days

getnumberweeksandhouses():

Global variables – Number\_of\_weeks and Number\_of\_houses

READ- DATA CWK SHOPPING DATA WEEK 1 FILE B

CREATE an empty list called Number\_of\_Houses\_confirmed

LOOP through each item in the file except the first item

IF to check it is already confirmed add it to Number\_of\_Houses\_confirmed list

Increment is added by 1

LOOP through each item in the second row  
 IF "WEEK" in header:  
The number of weeks will be incremented by 1

dataofshoppingList():

Global variable of Number\_of\_weeks

READ - DADSA CWK SHOPPING DATA WEEK 1 FILE A

LOOP through each row in store and the item of product and skip the first row

CREATE a new Item object using the data from the file

ADD the new Item object to the global Items list.

LOOP through the files headers

IF the current header contains the string “STORE”

LOOP through the rows in DataOfShoppingList

IF not on first row and the item is accessible at the store (containing “Y”)

ADD the object for the item stored in Items to the ItemsStocked list

ADD to make new store object

ADD the new Store to the global store list

getdataofHouses()

Global variable of Number\_of\_weeks and Number\_of\_Houses

READ- DATA CWK SHOPPING DATA WEEK 1 FILE B

LOOP through each column on the file

CREATE RequiredItemForHouse list

LOOP in range of the Number\_of\_weeks

LOOP through each row in data of houses

IF the current house wants any quantity of the current item

LOOP through the Item objects in Items

IF the item object name is the same as the current item name

ADD the item object and the quantity required to the correct week sub list in RequiredItemForHouse

CREATE house object and ADD it to the global Houses list.

**Main**

discoverbeststore(product\_of\_items):

CREATE a list of integers starting at zero. One for each store.

LOOP through the stores in Stores:

IF the item is listed Increment the current store StorePoints by 1

Finds the best store by getting the top number in the list and getting its index  
RETURN index

buyingitemofproducts(products\_of\_items, storeindex, week):

CREATE an empty list called missed\_items  
Store = Stores[StoreIndex]

LOOP through the items

IF listed look through the item is already in the stores ItemsBought list then Increment the current items quantity by the new items quantity.

IF not APPEND the item to the correct week sub list in the stores ItemsBought list.

ELSE APPEND the item to MissedItems list

RETURN the MissedItems list

BuyingAlternatives(Product\_of\_items, StoreIndex, week):

CREATE a copy of the product of items

LOOP Through Items

IF the available items are in stock in the store

IF listed check if the item is already in the stores ItemsBought list then Increment the current items quantity by the new items quantity.

APPEND the item to the correct week sub list in the stores ItemsBought list.

IF the available item is already in the store alternative list

IF the item exists then Increment

APPEND if the current item is not in the current store alternative list;

RETURN alternatives\_missed

bestalternatives(missed\_product\_of\_items):

CREATE empty list of Alternative\_Items

LOOP through each missed\_product of items

LOOP through each item in the Item global variable

ADD a list of all the words that are in both the item and the item

IF statement by using **len** to see how much characters matched

ADD the final best alternative for the current item missed to the alternatives list.

RETURN AlternativeItems.

shoppingalgo():

global variable of day\_week\_counter

LOOP through the number of weeks

LOOP through each house in the Houses list

CREATE variables (visit, count and day\_Week\_counter) as counters to track values added

CALL ItemsRequired.

CALL DiscoverBestStore in ItemsRequired

CALL BuyingItemOfProducts in ItemsRequired

CALL DiscoverBestStore passing in Missed\_Items.

CALL BuyingItemOfProducts passing in Missed\_Items,

SET the shops dayVisited to the day and Increment count by 1

ELSE begin alternative items by calling BestAlternatives passing Missed\_Items. SET alternatives equal to the list returned.

SET the house.DayofDelivery for the current week as count.

**Print statements**

PRINT("Here is the Shopping Schedule: ")  
PRINT what week you in

PRINT The day of the store you should visit

PRINT the quantity of items, price of it and the name of item

PRINT out the alternative item

Increment day\_week\_counter += 1

Day\_week\_counter = 0

PRINT("Here is the Delivery Schedule: ")

PRINT what week you in

PRINT(f" {daynames()[Day\_week\_counter]}") – the list of days of the week will be printed

PRINT which houses should be delivered

ELSE

PRINT("No need to deliver today!")

Increment day\_week\_counter += 1

Day\_week\_counter = 0

**Diagrams**

Diagram

Description automatically generated

Diagram

Description automatically generated

**Task 2**

**Import packages**

IMPORT CSV

**Classes**

Class Store:

FUNCTION INITIALISE Attributes of Store Object

(self, tag, stockitem, numberofweeks):

self.Tag = tag  
self.StockItem = stockitem  
self.Itemsbought = []  
self.alternatives = []

self.dayVisited = 0

for i in range(numberofweeks):  
 self.Itemsbought.append ([])  
 self.alternatives.append ([])

availableitemsinstock(self, item)

BOOLEAN used to check IF the item is available in stock in the store

Class Item

FUNCTION INITIALISE Attributes of ITEM Object

\_\_init\_\_(self, nameofitem, price):

self.NameofItem = nameofitem  
self.Price = price

Class House

FUNCTION INITIALISE Attributes of House Object

\_\_init\_\_(self, housenumber, Itemsrequired, numberofweeks):

self.HouseNumber = housenumber  
self.ItemsRequired = Itemsrequired  
self.DayofDelivery = []  
for i in range(numberofweeks):  
 self.DayofDelivery.append([])

**Global Variables**

CREATE empty list called Stores – an empty list that contains Store objects   
CREATE empty list called Houses - an empty list that contains House objects  
CREATE empty list called Items - an empty list that contains Item objects  
  
Number\_of\_weeks = 0 – An integer to store the number of weeks   
Number\_of\_Houses = 0 – An integer to store the number of weeks

**Helper Functions**

readcsv(filepath):

CREATE an empty list called list\_of\_data

OPEN the csv file located at filepath

READ csv file  
LOOP through each for row in csvreader:  
APPEND the row to a list

RETURN list\_of\_data

daynames():

this is a list and will show the different days of the week

days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

RETURN days

getnumberweeksandhouses():

Global variables – Number\_of\_weeks and Number\_of\_houses

READ- DATA CWK SHOPPING DATA WEEK 4 FILE B

CREATE an empty list called Number\_of\_Houses\_confirmed

LOOP through each item in the file(2nd column instead of 1 due to cost column)

IF to check it is already confirmed add it to Number\_of\_Houses\_confirmed list

Increment is added by 1

LOOP through each item in the second row  
 IF "WEEK" in header:  
The number of weeks will be incremented by 1

dataofshoppinglist():

Global variable of Number\_of\_weeks

READ - DADSA CWK SHOPPING DATA WEEK 4 FILE A

LOOP through each row in store and the item of product and skip the first row

CREATE a new Item object using the data from the file

ADD the new Item object to the global Items list.

LOOP through the files headers

IF the current header contains the string “STORE”

LOOP through the rows in DataOfShoppingList

IF not on first row and the item is accessible at the store (containing “Y”)

ADD the object for the item stored in Items to the ItemsStocked list

ADD to make new store object

ADD the new Store to the global store list

getdataofhouses()

Global variable of Number\_of\_weeks and Number\_of\_Houses

READ- DATA CWK SHOPPING DATA WEEK 4 FILE B

LOOP through each column on the file

CREATE RequiredItemForHouse list

LOOP in range of the Number\_of\_weeks

LOOP through each row in data of houses

IF the current house wants any quantity of the current item

LOOP through the Item objects in Items

IF the item object name is the same as the current item name

ADD the item object and the quantity required to the correct week sub list in RequiredItemForHouse

CREATE house object and ADD it to the global Houses list.

**Main**

discoverbeststore(product\_of\_items):

CREATE a list of integers starting at zero. One for each store.

LOOP through the stores in Stores:

IF the item is listed Increment the current store StorePoints by 1

Finds the best store by getting the top number in the list and getting its index  
RETURN index

buyingitemofproducts(products\_of\_items, storeindex, week):

CREATE an empty list called missed\_items  
Store = Stores[StoreIndex]

LOOP through the items

IF listed look through the item is already in the stores ItemsBought list then Increment the current items quantity by the new items quantity.

IF not APPEND the item to the correct week sub list in the stores ItemsBought list.

ELSE APPEND the item to missed\_Items list

RETURN the MissedItems list

buyingalternatives(product\_of\_items, storeindex, week):

CREATE a copy of the product of items

LOOP Through Items

IF the available items are in stock in the store

IF listed check if the item is already in the stores ItemsBought list then Increment the current items quantity by the new items quantity.

APPEND the item to the correct week sub list in the stores ItemsBought list.

IF the available item is already in the store alternative list

IF the item exists then Increment

APPEND if the current item is not in the current store alternative list;

RETURN alternatives\_missed

bestalternativesatStore(missed\_product\_of\_items, storeIndex):

CREATE empty list of Alternative\_Items

GET the current store from the storeindex passed

LOOP through each missed\_product of items

LOOP through each productitem in the self.StockItem

ADD a list of all the words that are in both the item and the item

IF statement by using **len** to see how much characters matched

ADD the final best alternative for the current item missed to the alternatives list.

IF no alternatives found

ADD the price difference between the missed items price and the current items price

IF the price difference is smaller than the current smallest\_price\_difference

SET this priceDifference as the smallest price difference

SET the current item as the BestAlternative

APPEND the final best alternative for the current missed product of item to the alternatives list.

RETURN AlternativeItems.

shoppinglgo():

global variable of day\_week\_counter

LOOP through the number of weeks

LOOP through each house in the Houses list

CREATE variables (visit,count,day\_week\_counter) counters to track values added

CREATE empty StoreVisited array

CALL ItemsRequired.

CALL DiscoverBestStore in ItemsRequired

IF store in storesVisited is false

APPEND the store you visited and increment the visit count by 1

CALL DiscoverBestStore passing in Missed\_Items.

CALL BuyingItemOfProducts passing in Missed\_Items,

SET the shops dayVisited to the day and Increment count by 1

ELSE begin alternative items by calling BestAlternatives passing Missed\_Items. SET alternatives equal to the list returned.

CALL alternatives, store and the current week.

Set Missed items equal to the returned list.

SET the house.DayofDelivery for the current week as count.

**Print statements**

PRINT("Here is the Shopping Schedule: ")  
PRINT what week you in

PRINT The day of the store you should visit

PRINT the quantity of items, price of it and the name of item

PRINT out the alternative item

Increment day\_week\_counter += 1

Day\_week\_counter = 0

PRINT("Here is the Delivery Schedule: ")

PRINT what week you in

PRINT(f" {daynames()[Day\_week\_counter]}") – the list of days of the week will be printed

PRINT which houses should be delivered

ELSE

PRINT("No need to deliver today!")

Increment day\_week\_counter += 1

Day\_week\_counter = 0

**Diagrams**

Diagram

Description automatically generated

Diagram

Description automatically generated

I added 4 instead of 3 because there are now 4 stores(Store A, Store B, Store C and Cheap store) and 15 houses since task 1 is only 7

Diagram

Description automatically generated

**Task 3**

**Import packages**

IMPORT CSV

**Classes**

Class Store:

FUNCTION INITIALISE Attributes of Store Object

(self, tag, stockitem, numberofweeks):

self.Tag = tag  
self.StockItem = stockitem  
self.Itemsbought = []  
self.alternatives = []

self.dayVisited = 0

for i in range(numberofweeks):  
 self.Itemsbought.append ([])  
 self.alternatives.append ([])

availableitemsinstock(self, item)

BOOLEAN used to check IF the item is available in stock in the store

Class Item

FUNCTION INITIALISE Attributes of ITEM Object

\_\_init\_\_(self, nameofitem, price):

self.NameofItem = nameofitem  
self.Price = price

Class House

FUNCTION INITIALISE Attributes of House Object

\_\_init\_\_(self, housenumber, Itemsrequired, numberofweeks):

self.HouseNumber = housenumber  
self.ItemsRequired = Itemsrequired  
self.DayofDelivery = []  
for i in range(numberofweeks):  
 self.DayofDelivery.append([])

**Global Variables**

CREATE empty list called Stores – an empty list that contains Store objects   
CREATE empty list called Houses - an empty list that contains House objects  
CREATE empty list called Items - an empty list that contains Item objects  
  
Number\_of\_weeks = 0 – An integer to store the number of weeks   
Number\_of\_Houses = 0 – An integer to store the number of weeks

**Helper Functions**

readcsv(filepath):

CREATE an empty list called list\_of\_data

OPEN the csv file located at filepath

READ csv file  
LOOP through each for row in csvreader:  
APPEND the row to a list

RETURN list\_of\_data

daynames():

this is a list and will show the different days of the week

days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

RETURN days

getnumberweeksandhouses():

Global variables – Number\_of\_weeks and Number\_of\_houses

READ- DATA CWK SHOPPING DATA WEEK 7 FILE B

CREATE an empty list called Number\_of\_Houses\_confirmed

LOOP through each item in the file(2nd column instead of 1 due to cost column)

IF to check it is already confirmed add it to Number\_of\_Houses\_confirmed list

Increment is added by 1

LOOP through each item in the second row  
 IF "WEEK" in header:  
The number of weeks will be incremented by 1

dataofshoppinglist():

Global variable of Number\_of\_weeks

READ - DADSA CWK SHOPPING DATA WEEK 7 FILE A

LOOP through each row in store and the item of product and skip the first row

CREATE a new Item object using the data from the file

ADD the new Item object to the global Items list.

LOOP through the files headers

IF the current header contains the string “STORE”

LOOP through the rows in DataOfShoppingList

IF not on first row and the item is accessible at the store (containing “Y”)

ADD the object for the item stored in Items to the ItemsStocked list

ADD to make new store object

ADD the new Store to the global store list

getdataofhouses()

Global variable of Number\_of\_weeks and Number\_of\_Houses

READ- DATA CWK SHOPPING DATA WEEK 7 FILE B

LOOP through each column on the file

CREATE RequiredItemForHouse list

LOOP in range of the Number\_of\_weeks

LOOP through each row in data of houses

IF the current house wants any quantity of the current item

LOOP through the Item objects in Items

IF the item object name is the same as the current item name

ADD the item object and the quantity required to the correct week sub list in RequiredItemForHouse

CREATE house object and ADD it to the global Houses list.

**Main**

discoverbeststore(product\_of\_items):

CREATE a list of integers starting at zero. One for each store.

LOOP through the stores in Stores:

IF the item is listed Increment the current store StorePoints by 1

Finds the best store by getting the top number in the list and getting its index  
RETURN index

buyingitemofproducts(products\_of\_items, storeindex, week):

CREATE an empty list called missed\_items  
Store = Stores[StoreIndex]

LOOP through the items

IF listed look through the item is already in the stores ItemsBought list then Increment the current items quantity by the new items quantity.

IF not APPEND the item to the correct week sub list in the stores ItemsBought list.

ELSE APPEND the item to missed\_Items list

RETURN the MissedItems list

buyingalternatives(product\_of\_items, storeindex, week):

CREATE a copy of the product of items

LOOP Through Items

IF the available items are in stock in the store

IF listed check if the item is already in the stores ItemsBought list then Increment the current items quantity by the new items quantity.

APPEND the item to the correct week sub list in the stores ItemsBought list.

IF the available item is already in the store alternative list

IF the item exists then Increment

APPEND if the current item is not in the current store alternative list;

RETURN alternatives\_missed

bestalternativesatStore(missed\_product\_of\_items, storeIndex):

CREATE empty list of Alternative\_Items

GET the current store from the storeindex passed

LOOP through each missed\_product of items

LOOP through each productitem in the self.StockItem

ADD a list of all the words that are in both the item and the item

IF statement by using **len** to see how much characters matched

ADD the final best alternative for the current item missed to the alternatives list.

IF no alternatives found

ADD the price difference between the missed items price and the current items price

IF the price difference is smaller than the current smallest\_price\_difference

SET this priceDifference as the smallest price difference

SET the current item as the BestAlternative

APPEND the final best alternative for the current missed product of item to the alternatives list.

RETURN AlternativeItems.

shoppinglgo():

global variable of day\_week\_counter

LOOP through the number of weeks

LOOP through each house in the Houses list

CREATE variables (visit,count,day\_week\_counter) counters to track values added

CREATE empty StoreVisited array

CALL ItemsRequired.

CALL DiscoverBestStore in ItemsRequired

IF store in storesVisited is false

APPEND the store you visited and increment the visit count by 1

CALL DiscoverBestStore passing in Missed\_Items.

CALL BuyingItemOfProducts passing in Missed\_Items,

SET the shops dayVisited to the day and Increment count by 1

ELSE begin alternative items by calling BestAlternatives passing Missed\_Items. SET alternatives equal to the list returned.

CALL alternatives, store and the current week.

Set Missed items equal to the returned list.

SET the house.DayofDelivery for the current week as count.

**Print statements**

PRINT("Here is the Shopping Schedule: ")  
PRINT what week you in

PRINT The day of the store you should visit

PRINT the quantity of items, price of it and the name of item

PRINT out the alternative item

Increment day\_week\_counter += 1

Day\_week\_counter = 0

PRINT("Here is the Delivery Schedule: ")

PRINT what week you in

PRINT(f" {daynames()[Day\_week\_counter]}") – the list of days of the week will be printed

PRINT which houses should be delivered

ELSE

PRINT("No need to deliver today!")

Increment day\_week\_counter += 1

Day\_week\_counter = 0

**Diagrams**

**Diagram

Description automatically generated**

Diagram

Description automatically generated