Return the best shop list for each individual household

Read the item and store it as a linked list data type

def setItem(item_linked)

ROW[1]

ROW[2]……..

	A	B	C	D	E	F
1	ITEM No.	NAME	COST (£)	STORE A	STORE B	STORE C
2		White Bread loaf large	£1.00	Y		Y
3		Brown Bread loaf large	£1.20		Y	
4		Bread Sliced White large	£0.60	Y	Y	Y
5		Bread Sliced Brown large	£0.70	Y	Y	

```
OPEN CSV AS READER
NEXT(READER)
FOR CSV ROW IN READER:
      NAME = ROW[1]
      PRICE = ROW[2]
      Store_list = []
      IF ROW[3] != EMPTY:
            Append store name into store_list
      IF ROW[4] != EMPTY:
            Append store name into store_list
      IF ROW[5] != EMPTY:
            Append store name into store_list
      Create a class item with name price and store list
      Create a linked list and store item class one by one

Return item
```

Set all the the information of each household into class

Read the shopping list and store it as a regular class

```
def setShoppingList(item_list,num):
```

A	B	C	D	E	F	G	H	I	
HOUSE NU	1A	1B	1C	1D	1E	2C	3E	1A	1B
PRODUCTS	QUANTITY WEEK 1							QUANTITY WEE	
White Brea	1						2	1	
Brown Bre	1	1	1	2	2	1	2	1	
Bread Sliced White la		1			2	2			

```
OPEN CSV AS READER
Shopping_list = []
Houeholds = FIRST LINE OF READER [1:num_household+1]
NEXT(READER) 2 times
IF num(week) Is 1:
      First column is 1
      Last column is num_household +1
ELIF num(week) Is 2:
      First column is 1+num_household
      Last column is num_household*2+1
ELSE:
      PRINT ERROR
Hid(Household ID) = 0
FOR i IN RANGE(FIRST COLUMN , LAST COLUMN):
      num_row = 0
      Create a class ShoppingList
      temp_shoppinglist = class ShoppingList(household_name)
      FOR ROW IN READER:
            If ROW[i] IS NOT EMPTY:
                  Create ItemQuantity class with Item class and Quantity
                  Temp_shoppinglist.optimise_item_list.add(ItemQuantity)
```

```
            Num_row += 1
      Reverse the item_list in temp_shoppinglist with reverse()
      Append temp_list into shoppinglist
      SEEK(0)
      NEXT()
      NEXT()
RETURN shoppinglist
```

Clean up the shopping list item requirement

Find the best permutation shop list for each household

```
def optimise_list(shop_list,item_linked):

permutation_shop_list = permutation("ABC")
lowest_sub = 10
lowest_p = permutation_shop_list[0]

FOR p in permutation_shop_list:
      Substitute = 0
      Current = linked list head
      WHILE CURRENT NOT EQUAL NONE
            CHECK THE PERMUTATION MATCH THE SHOP REQUIREMENT
            IF NOT
                  Substituite += 1
      IF substitute SMALLER THAN lowest_sub:
            Lowest_sub = substitute
            Lowest_p = p
IF lowest_sub NOT EQUAL 0:
      Substituition(shop_list,item_linked,lowest_p)
RETURN lowest_p
```

Substitute the shoppingList item with the best permutation shop list

Substitute the item if the item cannot be bought in the best permutation shop list

```
def substituition(shop_list,item_list,best_permutation)

current = shop_list.item linked list head
WHILE CURRENT NOT EQUAL NONE
       IF CURRENT ITEM STORE NOT MATCH THE best_permutation:
            Item = item_list.search(current.data.item.name)
            Prev_item = item before Item
            Next_item = item after Item
            Prew_w = 0
            Next_w = 0
            FOR WORD IN item.data.name
                  IF PREV_ITEM WORD COUNT MORE THAN NEXT_ITEM
                        Prew_w += 1
                  ELSE IF PREV_ITEM WORD COUNT LESS THAN NEXT_ITEM
                        Next_W += 1
            IF PREW_W BIGGER THAN NEXT_W
                  Set the Item to PREV_ITEM
            ElSE IF NEXT_W BIGGER THAN PREW_W
                  Set the Item to NEXT_ITEM
            ELSE:
                  PRINT ERROR
      CURRENT = CURRENT.get_next()
```

Pick the possible day of delivery and shopping to fulfilled the requirement

```
def delivery_date(best_permutation):


delivery_date = []
FOR P in permutation(“ABC”)
      temp_list = best_permutation
      num_done = 0
      extra_day = p
      FOR bp in temp_list
            Complete = False
            IF BP IS IN p(PERMUTAION):
                  Complete = True
                  Num_done += 1
            Else:
                  IF BP HAS SHOP IN LAST SHOP IN EXTRA_DAY:
                        Complete = True
                        Num_done += 1
                  ELSE IF
                        FOR DAY IN “ABC”
                              IF DAY EQUAL BP LAST REQUIRMENT
                                    Complete = True
                                    Num_done += 1
                                    Extra_day += DAY
                  IF COMPLETE EQUAL False
                        Extra_day += bp
                        Num_done += 1
```

```
                        Complete = True
            IF NUM_DONE EQUAL NUM OF HOUSEHOLD
                  Append extra_day into delivery_date
DELIVERY_DATE = THE SMALLEST LEN IN THE DELIVERY DATE
RETURN DELIVERY_DATE
```

BUY THE ITEM AND DELIVER

```
Def
delivery(best_delivery_date,ship_list,best_permutation,item_dict_,it
em_price_dict):


Num_day_buy = len(best_delivery_day)
Bdd = best_delivery_day
Bp = best_permutation(list)
Shopping_schedule = []
Households_delivery_day = []


FOR I IN RANGE(num_day_buy)
      Temp_item_dict = item_dict.copy()
      Temp_hdd = []
      FOR num_household,household IN ENUMERATE(shop_list):
            Current_day = bp[num_household].count(bdd[i]
            IF I EQUAL LAST DAY
                  Next_day = 1
            Else:
                  Next_day = bp[num_household].count(bdd[i+1])
            IF THE HOUSEHOLD NEED THE CURRENT DAY AND NEXT DAY SHOP ITEM
                  Current = household.optimised_item_list.head(linked
list)
                  WHILE CURRENT IS NOT EQUAL TO NONE
                        NEXT_CURRENT = CURRENT.get_next()
                        IF CURRENT STORE CONTAIN bdd[i]
```

Item.dict is a dictionary about item_name and quantity requirement

```
                                Temp_item_dict[current_item_name] +=
current.quantity
                                REMOVE THE CURRENT ITEM FROM LINKED LIST
                          Current = next_current
                    IF LINKED_LIST EMPTY:
                          Temp_hdd.append(HOUSEHOLD NAME)
              Shopping_schedule insert temp_item_dict


PRINT OUT THE RESULT
```