Project Report

1. Introduction

This Project has several warehouses located all around warehouses, each warehouse has different capacity.

However, there is no system connected all the warehouse for the smart system. We decided to design the program to make the warehouse system an automatic storage system by following the requirement design a program such that.

- When received an order, a robot will pick up an item from a warehouse and transfer.
- When received an order, the belt will output 1 item at a time.
- When received a command, a robot will store an item at a specific location.

Based on these requirements, we build a lot of storage space. It will support product relocation between warehouses. It is packed into the belt to move. And these program can store products from the outside to the warehouse or to remove items from the warehouse easily. So you can find the exact position of the product that you want to.

2. Requirement Analysis

Functional Requirement and specification

From the overview we will be divided into three parts: belt, warehouses, input & output Command

1. Warehouse Specifications requirement

A row is a 2-dimensional grid, each space in a grid is used to store an item, each warehouse has a robot to pick up and store items. There are 5 warehouses, warehouse1 connects with a conveyor belt, Warehouse 2 and Warehouse 3, Warehouse 2 connects with Warehouse 4 and 5, and There is a robot running around to transfer items from a warehouse to a conveyor belt. The conveyor belt can hold up to 10 items. All warehouses can storage 9675 products.

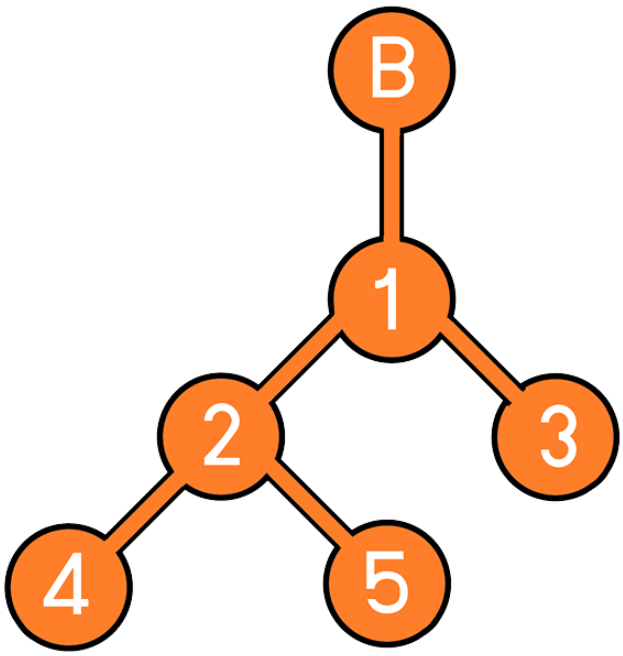
Warehouse 1, 2, and 3 have 5 rows of 10x10 grid. Can storage 500x3 = 1500 products.

Warehouse 4 has 7 rows of 5x5 grid. Can storage 175 products.

Warehouse 5 has 20 rows of 20x20 grid. Can storage 8000 products.

- When received an order, a robot will pick up an item from a warehouse and transfer.
- When received an order, the belt will output 1 item at a time.
- When received a command, a robot will store an item at a specific location.

2. Belt



2.2 Product ID requirement

Each product has a unique id in a form of 4 characters: x y z

- x represents a type of the item. It has a value of A to Y.
- y represents a row number of the item. It has a value of 1 to 5.
- z represents a row number of the item. It has a value of 00 to 99.

2.3 Input Command requirement

There are several commands we can give to the system. The commands have following formats.

- 0XXXX Retrieve a product id XXXX
- 1XXXX Store a product id XXXX
- 2XY00 Sort Warehouse X at row Y
- 30000 Retrieve a product from the conveyor belt
- 40000 Output information of all warehouses

- 5XXXX Search for a product ID XXXX
- 9XXXXYYYY Manually put a product id XXXX at position YYYY

Non-Functional Requirement

Output command requirement

- Retrieving command: 0XXXX

If the system can operate successfully, the system should print out the following statements in this order:

- Retrieving Successfully.

If belt is full, the system should print out following statements:

- Belt is full. Cannot retrieve the product

If slot is empty, the system should print out following statements:

- Slot is empty. Cannot retrieve the product.

If the product is still in the belt, the product can't be retrieved. The system will display that.

- now product XXXX is on belt.

- Storing command: 1XXXX

If the system can operate successfully, the system should print out the following statements in this order:

- Moving from Belt to A
- Moving from A to C
- Storing a product id XXXX in warehouse C: row y slot z
- Moving from C to A
- Moving from A to Start
- Storing Successfully!

If slot is unavailable, the system should print out following statements in this order:

- Slot is occupied. Can't store the product.

If the product is already stored and still in the warehouse, the system will display.

o product has been stored.

If the product is still in the belt, it can't be stored.

o now product XXXX is on belt.

- Sorting command: 2XY00

If the system can operate successfully, the system should print out the following statements in this order:

- Sorting process for warehouse A is complete.

- Retrieving from the belt command: 30000

If the system can operate successfully, the system should print out the following statements in this order:

- Retrieve a product with id XXXX from the belt.
- The belt now has x products on the line.

If there is nothing on the belt, the system should print out following statements in this order:

- The belt is empty. Cannot retrieve the product from the belt.

- Warehouse Information: 40000

If the system can operate successfully, the system should print out the following statements in this order:

- Warehouse A
- Numbers of Rows: 5
- Numbers of total products: 8
- Product in row 1: id A100, C108, E111
- Product in row 2: id –
- Product in row 3: id L355
- Product in row 4: id Q450
- Product in row 5: id U500, W501,

- Searching for a product: 5XXXX

If the system can operate successfully, the system should print out the following statements in this order:

- Found the product at XXXX.

If the system cannot find the product, the system should print out following statements in this order:

- Product not found.

- Manually moving the product: 9XXXXYYYY

If the system can operate successfully, the system should print out the following statements in this order:

- Move product XXXX to YYYY

If the slot is occupied, the system should print out following statements in this order:

- Slot is occupied. Failed to move.

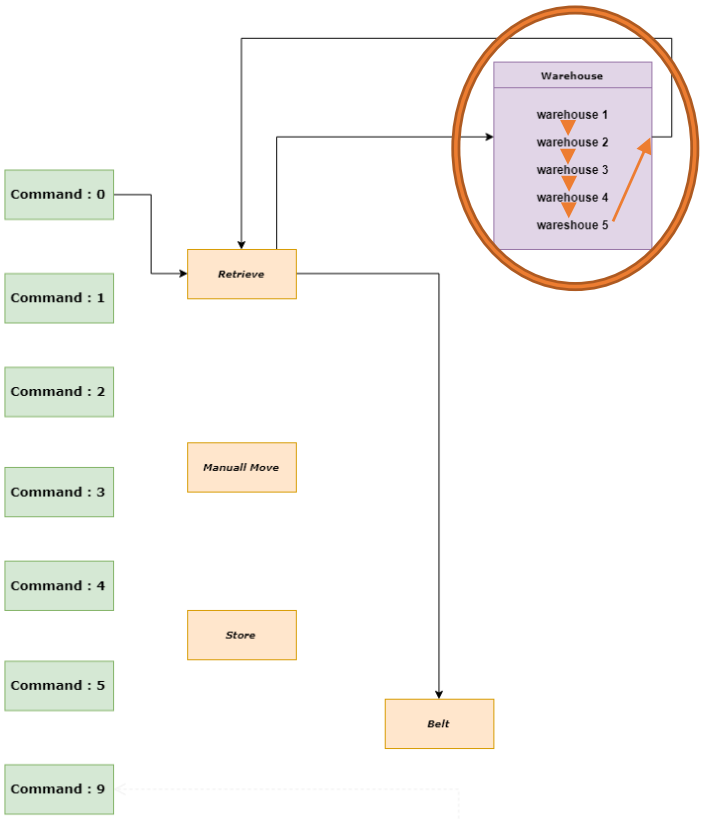
If the product is still in the belt, the product can't be moved.

- o now product XXXX is on belt.

If function 9XXXXYYYY only has XXXXX but not has YYYY it will not work.

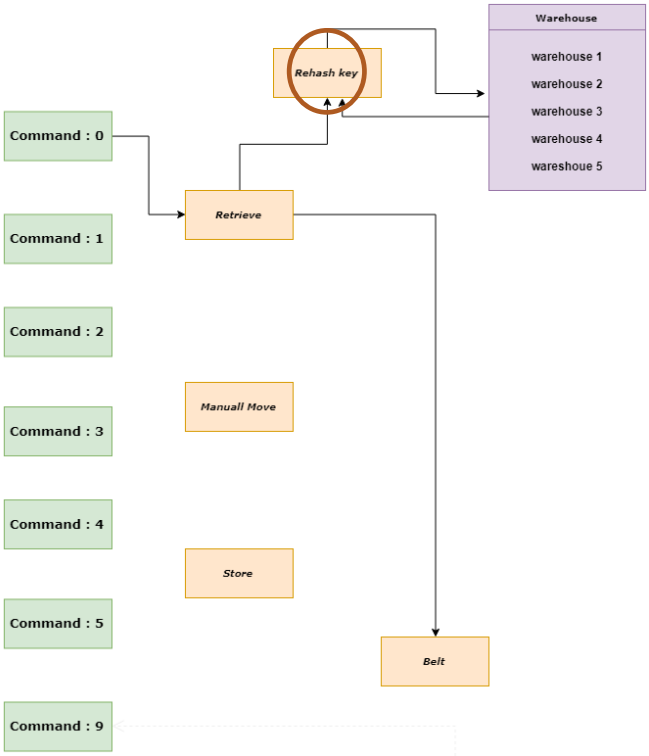
- Error: please input positon YYYY

Rehash key



(Normal way)

Normally when we need find some of product. We will need to find every warehouse by warehouse to find it.



(Rehash)

(Concept how to use rehash to retrieving product and storage)

It's key to solve how to searching, re-check, retrieving, storage, Manually

but if we use Rehash we build key shortcut to find all the warehouse by searching in key shortcut in easily and faster than normally way.

Key shortcut it will store the current product id + position. each warehouse is a grid map to find it more convenient to search where it is, (order of one)

4. Offers

-Lower case If the product name is given by a to y, the system will change to A to Y.

It is required that characters 2 and 6 must be English characters. The rest must be numbers ,by follow product id requirement .

- If the command entered is not equal to 5 or 9, the system will be warned.

"Your order is not clear please input again"

-Function hashing_key in the class Robot is a function to enter and decode. The product ID A100-Y599 is changed to 32600-45099, then the product id should be located in the warehouse, whichever slot, and also find out what position to store product id and store it. dictionary, because it wants to search product id as search as O (1)

Extra Code:

InputComman: 60000

Show all Product key in all warehouse

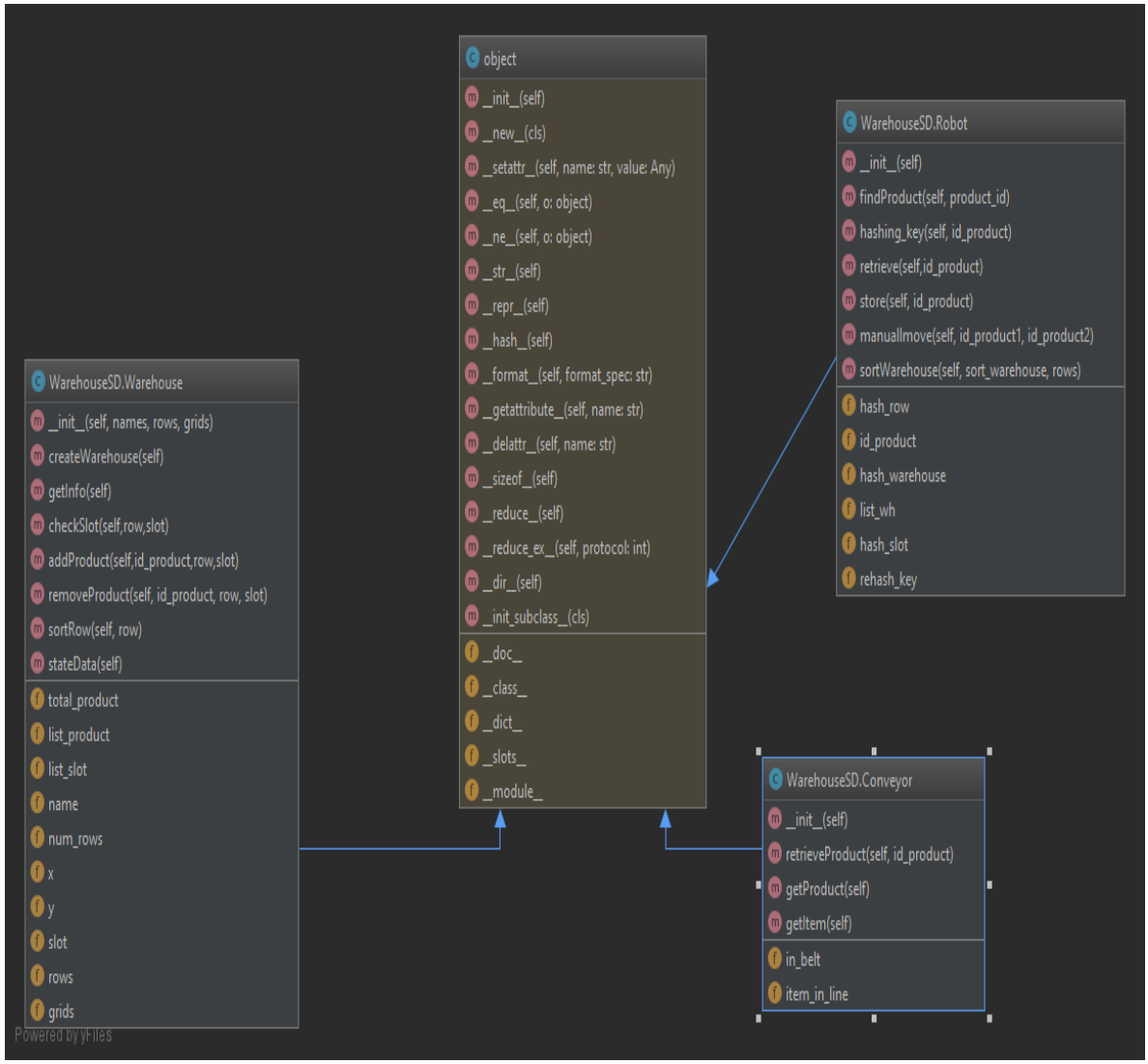
InputComman: 70000

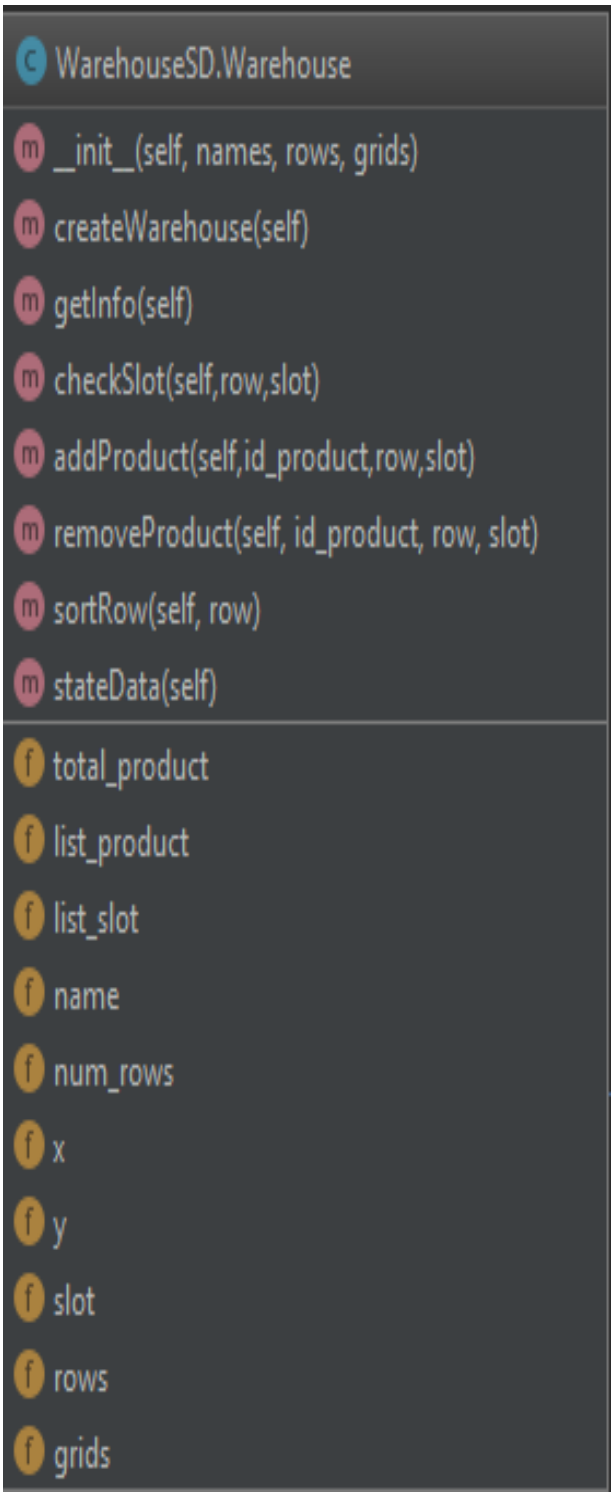
Show in side of all warehouse

Design

Design class warehouse

Relation between class_warehouse





Class Conveyor

Warehouse.createWarehouse ()

confirm statement to create the warehouse.

Warehouse.getInfo ()

View the warehouse information.

Warehouse.checkSlot (row, slot)

View the location in the warehouse to see if the slot in the row is empty.

Warehouse.addProduct (id_product, row, slot)

Put the product into the warehouse as a row slot.

Warehouse.removeProduct (id_product, row, slot)

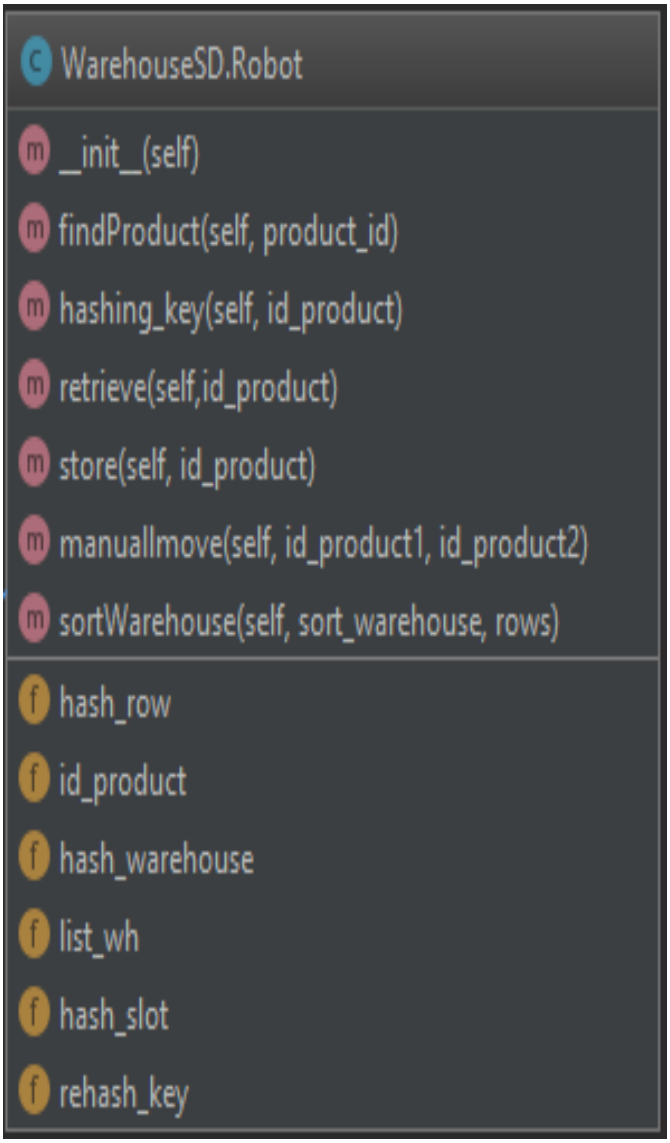
Remove the product from the warehouse as a row slot.

Warehouse.sortRow (row)

Take the product in the row and sort it out so that the product in the row goes back to where it should be.

Warehouse.stateData ()

Show all the warehouse



Class Robot

Robot.findProduct (product_id)

Is a function where the product id is located.

Robot.hashing_key (id_product)

A function that encodes and decodes to find the position or product id.

Robot.retrieve (id_product)

Function to remove product id from the warehouse.

Robot.store (id_product)

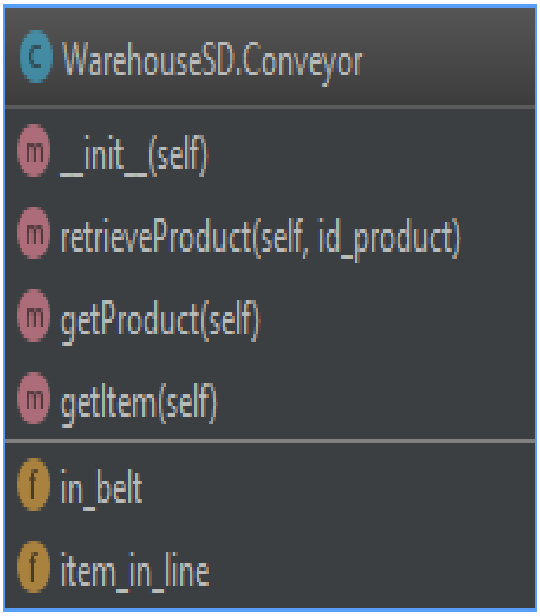
It's a function for putting a product into the warehouse where it should be.

Robot.manuallmove

It is a function to move a product that exists in the warehouse. If it is not, it will be added to it.

Robot.sortWarehouse

The function is to sort the required warehouse by specifying the row that you want to sort by the product in the row, through the sortRow of the Warehous class.



Class Conveyor

Conveyor.retrieveProduct (id_product)

Bring product id back to the belt.

Conveyor.getProduct ()

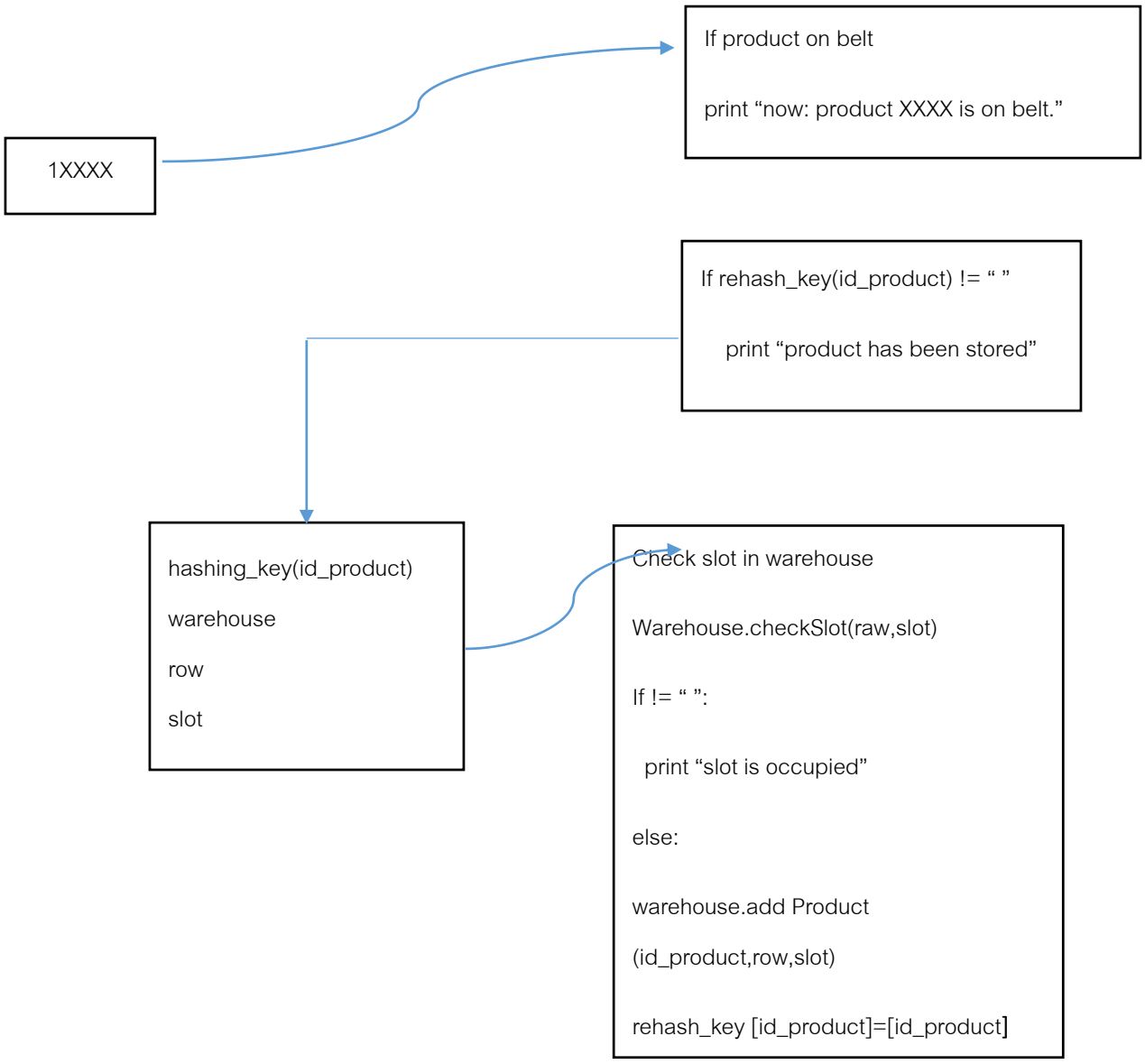
Take the product out of the belt as a first come first serve basis.

Conveyor.getItem ()

Browse product number in belt

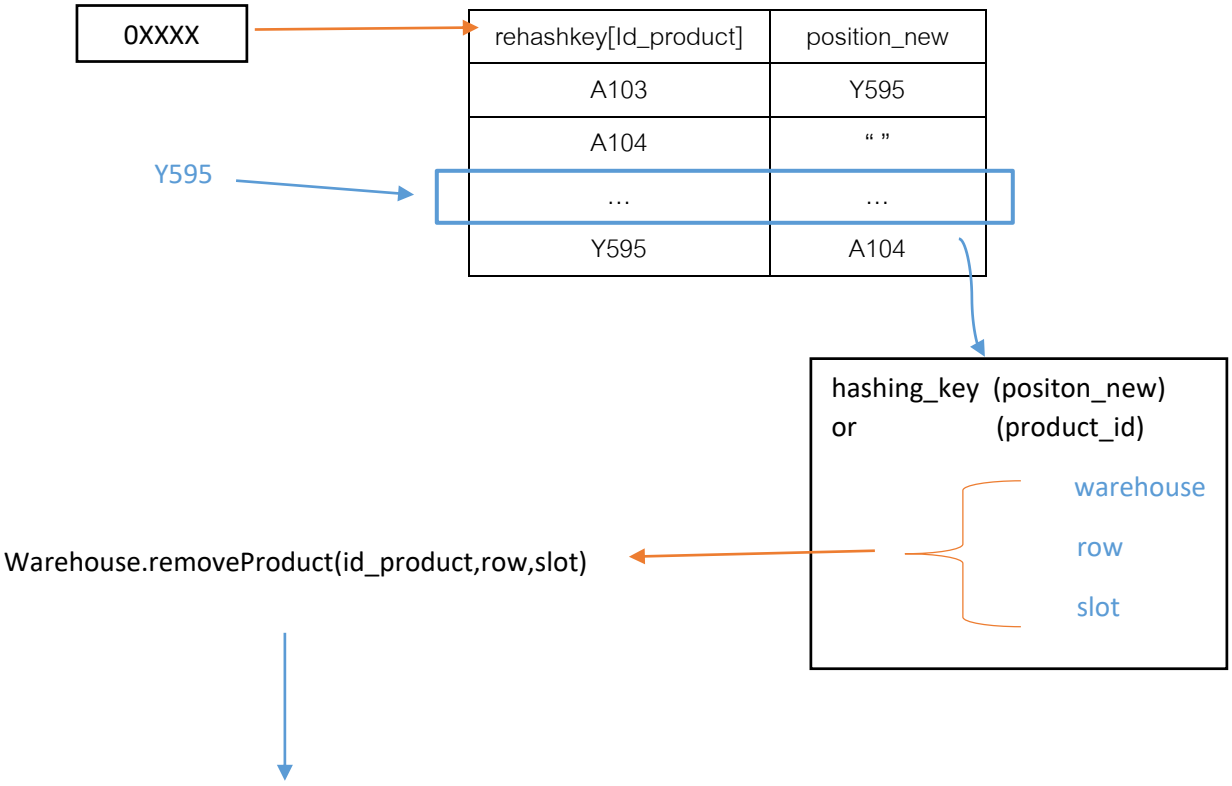
1XXXX Store product

Check hashing_key[id_product]. Check product on belt. If belt has product it will print “product XXXX is on belt”. If product has position it will print “now: product has been stored”. Check slot in warehouse. If row and slot in warehouse is occupied it will print “Slot is occupied” but if row and slot is not in warehouse it will add product by Warehouse.add() function and rehash id product.



0XXXX Retrive product

Check rehash_key[id_product]. If empty print “slot is empty. Cannot retrieve the product” else hashing_key(from position_new of product) because we don’t know where is it now remove product from warehouse to belt. New product on the belt (use function belt.retrieve(product_id)).



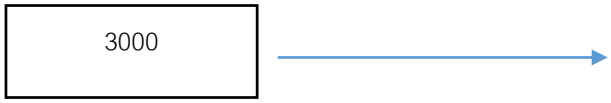
Robot take product to belt

rehash_key[id_product] = “Belt”

belt.retrieve(id_product)

30000

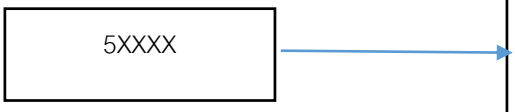
Check product in belt if count of product in belt = 0 print “The belt is empty” else print “Retrievefirst product from belt” rehash_key[first product in belt] = “” . Delete first product in belt. delete 1 count of product and print “count of product in line”




```
if self.item_in_line == 0:
    print (The belt is empty)
else:
    print ("Retrieve” +self.in_belt[0])
    r.rehash_key[self.in_belt[0]] = "    "
    del self.in_belt[0]
    self.item_in_line -= 1
    print (“the belt now has”+ str(self.item_in_line))
```

5XXXX search of product

Check if r.rehash_key[product_id] not equal empty print “position of product” else print “not found”

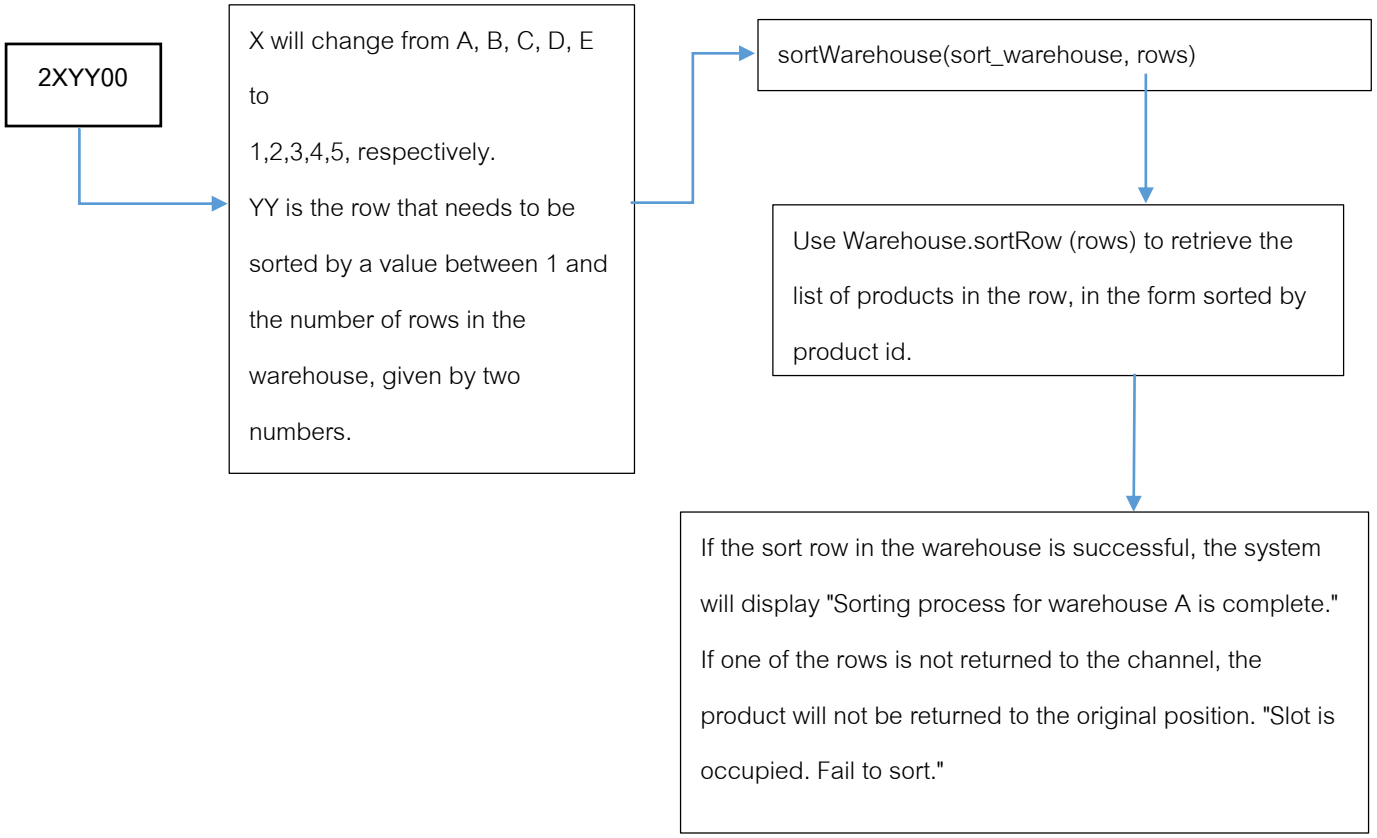



```
if r.rehash_key[product_id1] != "    ":
    print("Found the product at " +r.rehash_key[product_id1])
else:
    print("Product not found")
```

2XYY00

X is the warehouse you want to sort with names A, B, C, D, E.

YY is the row that needs to be sorted by a value between 1 and the number of rows in the warehouse, given by two numbers. If the sort row in the warehouse is successful, the system will display "Sorting process for warehouse A is complete." If one of the rows does not return to the slot, the product will not be returned to the original slot. "Slot is occupied. Fail to sort."



9XXXXYYYY

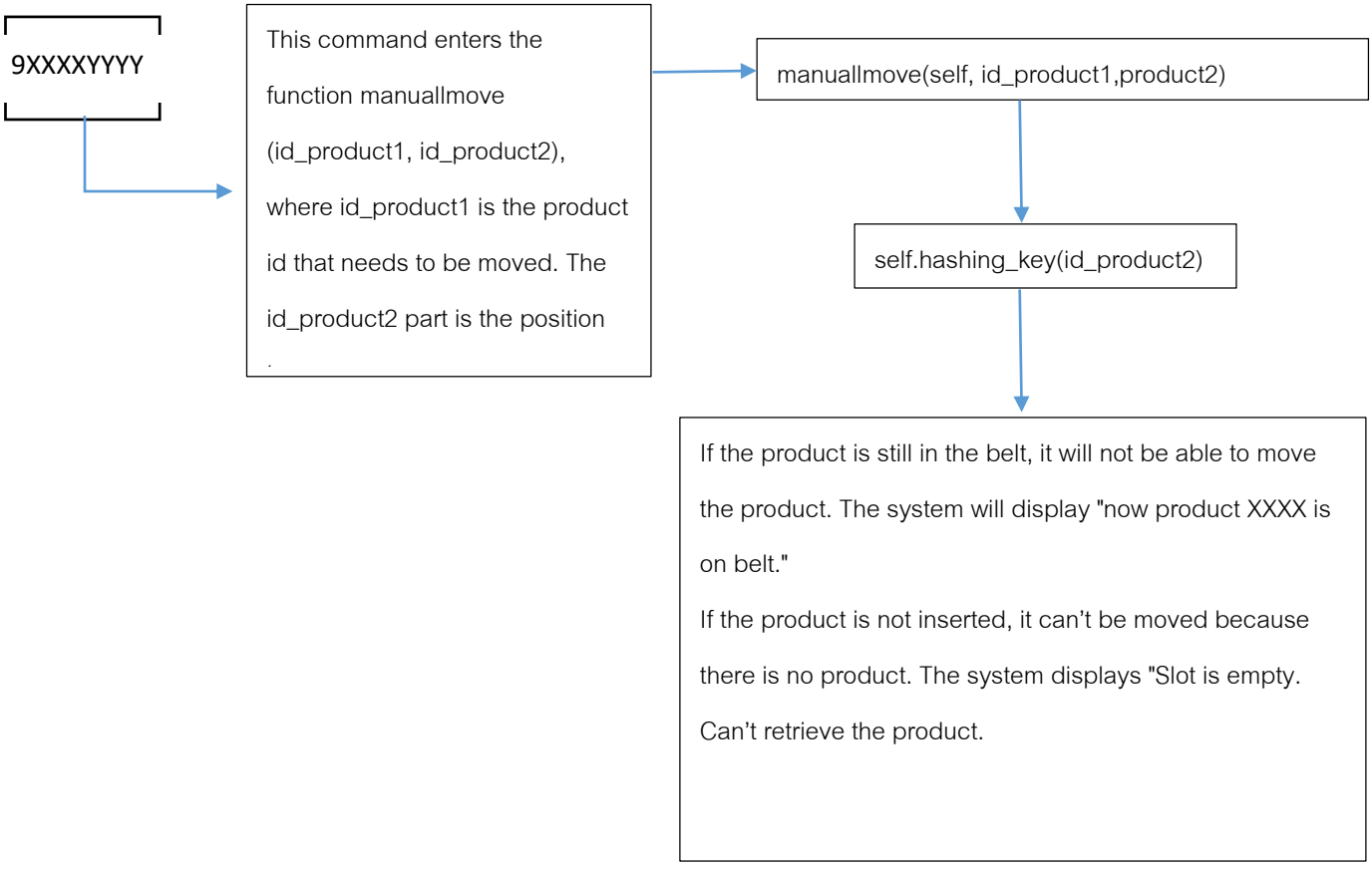
This command enters the function manuallmove (id_product1, id_product2), where id_product1 is the product id that needs to be moved. The id_product2 part is the position to move. The procedure is to put the position into the hashing_key, then check to see if the position is in the warehouse row slot empty, and then put the product id into the hashing_key function to see if the product id has been inserted. And not in the belt.

If the product can be moved to the desired position, the system will display "Move product XXXX to YYYY".

If the position is moved, the system will display "Slot is occupied. Failed to move."

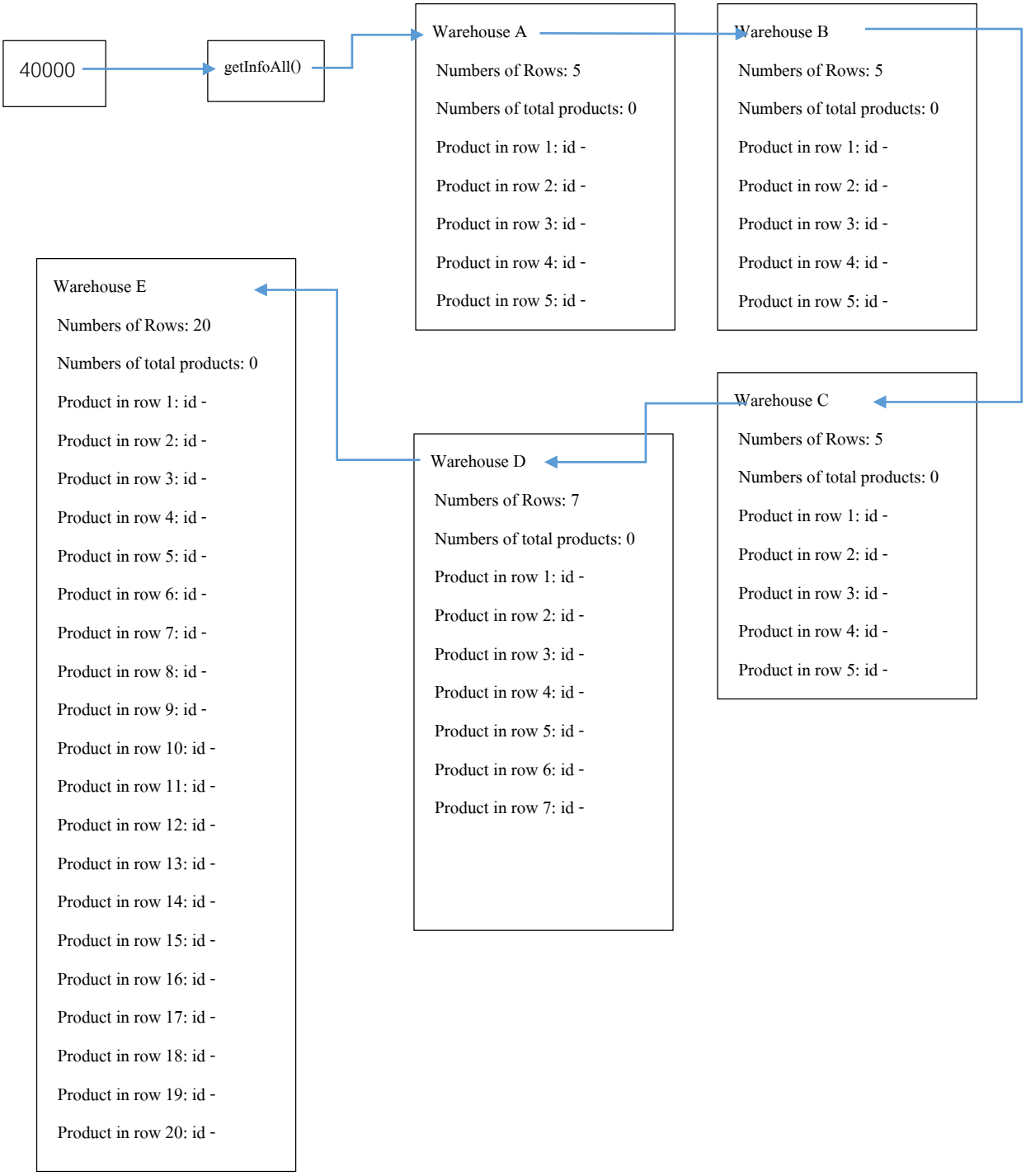
If the product is still in the belt, it will not be able to move the product. The system will display "now product XXXX is on belt."

If the product is not inserted, it can't be moved because there is no product. The system displays "Slot is empty. Can't retrieve the product.



40000

The system will display the data in every warehouse. It will display the names of the number of rows, the number of items in each row, through the function getInfoAll () to display all the warehouse data.



Design in Warehouse

All warehouses can storage 9675 products. (Example extra code 70000)

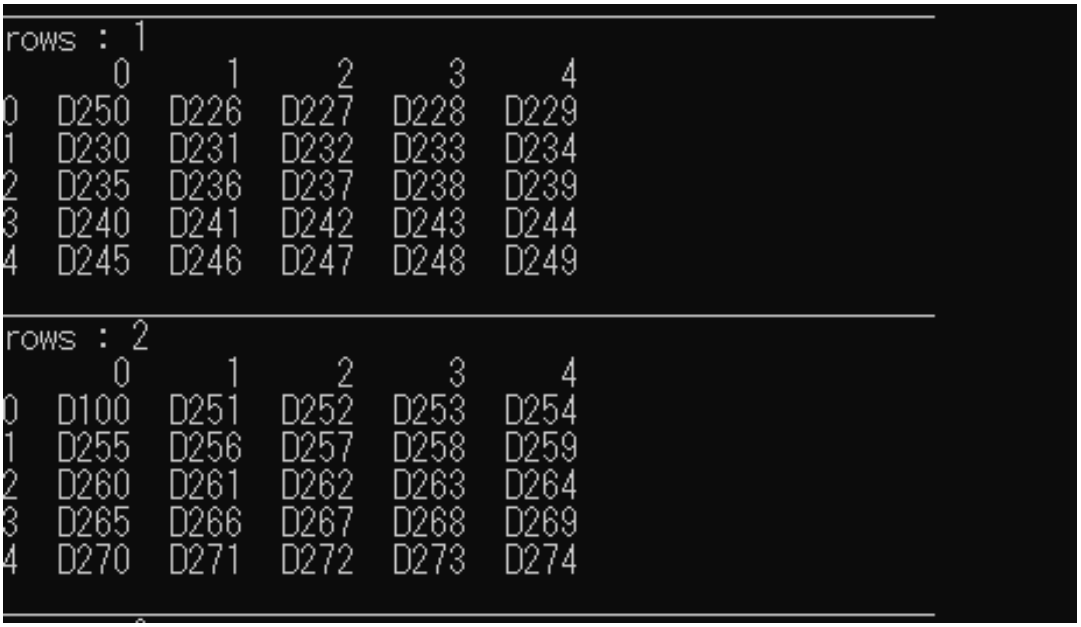
Empty warehouse

```
rows : 19
      0     1     2     3     4     5     6     7     8   ...    11    12    13    14    15    16    17    18    19
0                                                         ...
1                                                         ...
2                                                         ...
3                                                         ...
4                                                         ...
5                                                         ...
6                                                         ...
7                                                         ...
8                                                         ...
9                                                         ...
10                                                        ...
11                                                        ...
12                                                        ...
13                                                        ...
14                                                        ...
15                                                        ...
16                                                        ...
17                                                        ...
18                                                        ...
19                                                        ...

[20 rows x 20 columns]
rows : 20
      0     1     2     3     4     5     6     7     8   ...    11    12    13    14    15    16    17    18    19
0                                                         ...
1                                                         ...
2                                                         ...
3                                                         ...
4                                                         ...
5                                                         ...
6                                                         ...
7                                                         ...
8                                                         ...
9                                                         ...
10                                                        ...
11                                                        ...
12                                                        ...
13                                                        ...
14                                                        ...
15                                                        ...
16                                                        ...
17                                                        ...
18                                                        ...
19                                                        ...

[20 rows x 20 columns]
```

Sort product to warehouse



```
rows : 1
      0     1     2     3     4
0  D250  D226  D227  D228  D229
1  D230  D231  D232  D233  D234
2  D235  D236  D237  D238  D239
3  D240  D241  D242  D243  D244
4  D245  D246  D247  D248  D249

rows : 2
      0     1     2     3     4
0  D100  D251  D252  D253  D254
1  D255  D256  D257  D258  D259
2  D260  D261  D262  D263  D264
3  D265  D266  D267  D268  D269
4  D270  D271  D272  D273  D274
```

```
rows : 1
      0     1     2     3     4     5     6     7     8     9
0  A100  A101  A102  A103  A104  A105  A106  A107  A108  A109
1  A110  A111  A112  A113  A114  A115  A116  A117  A118  A119
2  A120  A121  A122  A123  A124  A125  A126  A127  A128  A129
3  A130  A131  A132  A133  A134  A135  A136  A137  A138  A139
4  A140  A141  A142  A143  A144  A145  A146  A147  A148  A149
5  A150  A151  A152  A153  A154  A155  A156  A157  A158  A159
6  A160  A161  A162  A163  A164  A165  A166  A167  A168  A169
7  A170  A171  A172  A173  A174  A175  A176  A177  A178  A179
8  A180  A181  A182  A183  A184  A185  A186  A187  A188  A189
9  A190  A191  A192  A193  A194  A195  A196  A197  A198  A199

rows : 2
      0     1     2     3     4     5     6     7     8     9
0  A200  A201  A202  A203  A204  A205  A206  A207  A208  A209
1  A210  A211  A212  A213  A214  A215  A216  A217  A218  A219
2  A220  A221  A222  A223  A224  A225  A226  A227  A228  A229
3  A230  A231  A232  A233  A234  A235  A236  A237  A238  A239
4  A240  A241  A242  A243  A244  A245  A246  A247  A248  A249
5  A250  A251  A252  A253  A254  A255  A256  A257  A258  A259
6  A260  A261  A262  A263  A264  A265  A266  A267  A268  A269
7  A270  A271  A272  A273  A274  A275  A276  A277  A278  A279
8  A280  A281  A282  A283  A284  A285  A286  A287  A288  A289
9  A290  A291  A292  A293  A294  A295  A296  A297  A298  A299
```

```
rows : 19
      0     1     2     3     4     5     6     7     8   ...    11    12    13    14    15    16    17    18    19
0   T300  T301  T302  T303  T304  T305  T306  T307  T308  ...  T311  T312  T313  T314  T315  T316  T317  T318  T319
1   T320  T321  T322  T323  T324  T325  T326  T327  T328  ...  T331  T332  T333  T334  T335  T336  T337  T338  T339
2   T340  T341  T342  T343  T344  T345  T346  T347  T348  ...  T351  T352  T353  T354  T355  T356  T357  T358  T359
3   T360  T361  T362  T363  T364  T365  T366  T367  T368  ...  T371  T372  T373  T374  T375  T376  T377  T378  T379
4   T380  T381  T382  T383  T384  T385  T386  T387  T388  ...  T391  T392  T393  T394  T395  T396  T397  T398  T399
5   T400  T401  T402  T403  T404  T405  T406  T407  T408  ...  T411  T412  T413  T414  T415  T416  T417  T418  T419
6   T420  T421  T422  T423  T424  T425  T426  T427  T428  ...  T431  T432  T433  T434  T435  T436  T437  T438  T439
7   T440  T441  T442  T443  T444  T445  T446  T447  T448  ...  T451  T452  T453  T454  T455  T456  T457  T458  T459
8   T460  T461  T462  T463  T464  T465  T466  T467  T468  ...  T471  T472  T473  T474  T475  T476  T477  T478  T479
9   T480  T481  T482  T483  T484  T485  T486  T487  T488  ...  T491  T492  T493  T494  T495  T496  T497  T498  T499
10  T500  T501  T502  T503  T504  T505  T506  T507  T508  ...  T511  T512  T513  T514  T515  T516  T517  T518  T519
11  T520  T521  T522  T523  T524  T525  T526  T527  T528  ...  T531  T532  T533  T534  T535  T536  T537  T538  T539
12  T540  T541  T542  T543  T544  T545  T546  T547  T548  ...  T551  T552  T553  T554  T555  T556  T557  T558  T559
13  T560  T561  T562  T563  T564  T565  T566  T567  T568  ...  T571  T572  T573  T574  T575  T576  T577  T578  T579
14  T580  T581  T582  T583  T584  T585  T586  T587  T588  ...  T591  T592  T593  T594  T595  T596  T597  T598  T599
15  U100  U101  U102  U103  U104  U105  U106  U107  U108  ...  U111  U112  U113  U114  U115  U116  U117  U118  U119
16  U120  U121  U122  U123  U124  U125  U126  U127  U128  ...  U131  U132  U133  U134  U135  U136  U137  U138  U139
17  U140  U141  U142  U143  U144  U145  U146  U147  U148  ...  U151  U152  U153  U154  U155  U156  U157  U158  U159
18  U160  U161  U162  U163  U164  U165  U166  U167  U168  ...  U171  U172  U173  U174  U175  U176  U177  U178  U179
19  U180  U181  U182  U183  U184  U185  U186  U187  U188  ...  U191  U192  U193  U194  U195  U196  U197  U198  U199

[20 rows x 20 columns]
rows : 20
      0     1     2     3     4     5     6     7     8   ...    11    12    13    14    15    16    17    18    19
0   E200  U201  U202  U203  U204  U205  U206  U207  U208  ...  U211  U212  U213  U214  U215  U216  U217  U218  U219
1   U220  U221  U222  U223  U224  U225  U226  U227  U228  ...  U231  U232  U233  U234  U235  U236  U237  U238  U239
2   U240  U241  U242  U243  U244  U245  U246  U247  U248  ...  U251  U252  U253  U254  U255  U256  U257  U258  U259
3   U260  U261  U262  U263  U264  U265  U266  U267  U268  ...  U271  U272  U273  U274  U275  U276  U277  U278  U279
4   U280  U281  U282  U283  U284  U285  U286  U287  U288  ...  U291  U292  U293  U294  U295  U296  U297  U298  U299
5   U300  U301  U302  U303  U304  U305  U306  U307  U308  ...  U311  U312  U313  U314  U315  U316  U317  U318  U319
6   U320  U321  U322  U323  U324  U325  U326  U327  U328  ...  U331  U332  U333  U334  U335  U336  U337  U338  U339
7   U340  U341  U342  U343  U344  U345  U346  U347  U348  ...  U351  U352  U353  U354  U355  U356  U357  U358  U359
8   U360  U361  U362  U363  U364  U365  U366  U367  U368  ...  U371  U372  U373  U374  U375  U376  U377  U378  U379
9   U380  U381  U382  U383  U384  U385  U386  U387  U388  ...  U391  U392  U393  U394  U395  U396  U397  U398  U399
10  U400  U401  U402  U403  U404  U405  U406  U407  U408  ...  U411  U412  U413  U414  U415  U416  U417  U418  U419
11  U420  U421  U422  U423  U424  U425  U426  U427  U428  ...  U431  U432  U433  U434  U435  U436  U437  U438  U439
12  U440  U441  U442  U443  U444  U445  U446  U447  U448  ...  U451  U452  U453  U454  U455  U456  U457  U458  U459
13  U460  U461  U462  U463  U464  U465  U466  U467  U468  ...  U471  U472  U473  U474  U475  U476  U477  U478  U479
14  U480  U481  U482  U483  U484  U485  U486  U487  U488  ...  U491  U492  U493  U494  U495  U496  U497  U498  U499
15  E100  E101  E102  E103  E104  E105  E106  E107  E108  ...  E111  E112  E113  E114  E115  E116  E117  E118  E119
16  E120  E121  E122  E123  E124  E125  E126  E127  E128  ...  E131  E132  E133  E134  E135  E136  E137  E138  E139
17  E140  E141  E142  E143  E144  E145  E146  E147  E148  ...  E151  E152  E153  E154  E155  E156  E157  E158  E159
18  E160  E161  E162  E163  E164  E165  E166  E167  E168  ...  E171  E172  E173  E174  E175  E176  E177  E178  E179
19  E180  E181  E182  E183  E184  E185  E186  E187  E188  ...  E191  E192  E193  E194  E195  E196  E197  E198  E199

[20 rows x 20 columns]
```