INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT010-3-1-FSD

FUNDAMENTALS OF SOFTWARE DEVELOPMENT

<<APU1F2006CS(IS) >>

HAND OUT DATE: 17TH AUGUST 2020

HAND IN DATE: 20TH SEPTEMBER 2020

WEIGHTAGE: 100%

STUDENT NAME	TP NUMBER
SIA DE LONG	TP060810

Table of Contents

1. Introduction

1.1 Problem Statement

There is an automobile manufacturing plant freshly built has been assembling few of different models of passenger cars. The plant has particular warehouse for each car models and each warehouse have few assembly sections for a particular model. All assembly sections will be using their required parts from their respective warehouse. Due to the economic slowdown, the company will only produce exactly one variant under each car model. The automobile manufacturing plant has decided to use an Automobile Parts Inventory Management System for all the warehouses. The inventory system has to be programmed in Python and the author had been recruited for the same.

1.2 Requirements

There are 4 main features which are compulsory to fulfil:

1.2.1 Parts Inventory Creation in Warehouses

This is a feature to record new part with its details while initial quantity of each parts and the assembly section they are used is important into text files in order to manage it. It must at least can stimulate 3 warehouses with at least 3 assembly sections and record at least 5 parts under each of them while supplier details for each of the part may be excluded.

1.2.2 Parts Inventory Update

This is a feature to update the quantity of parts either supplied from a suppliers or used in respective assembly section. Each part can be supplied by exactly one supplier only and can be provided to a particular assembly section only. However, one supplier can supply more than one part. The program is required to update at least 30% to 50% of parts either upon supplied from suppliers or after used to assembly sections and must check for available quantity in warehouse before granting request from respective assembly section. Lastly, it must be able to new parts inventory creation at any point of time.

1.2.3 Parts Inventory Tracking

This is a feature to track and display out every activities or status of parts in all warehouse. The inventory system must can display current total available quantity of all parts sorted in ascending order by part id, Records of all parts that has stock quantity less than 10 units by warehouse and also the parts and quantity used to respective assembly section by warehouse.

1.2.4 Searching Functionalities

This is a feature to let the users to search desired data from all the text files this system created and display it out. The inventory system must can display Part's record or its supplier details when searched by part's id and display all part details which supplied by supplier who selected by user.

1.3 Objective

1. To design an Automobile Parts Inventory Management System in Python which meet all the requirement and possibly adding any extra feature which is relevant and add value to the system.
2. To research on suitable parts will be using when assembly a car and possibly parts used when assembly different price of car.

1.4 Assumptions

1. Total parts will not exceed 10000, so that the id will be limited in 0000-9999.
2. Part ids are shared between every warehouse which mean same id cannot be used in another warehouse.
3. Part id should start with 'P' while supplier id starts with 'S'.
4. There is no need for graphics (user interface) in the program.
5. This program is only design for model Bios, Ambry and Barrier.
6. All warehouses will have the same assembly sections which are engine section, body work section and air-con section.
7. There will be no mistake when key-in data to the program.
8. There should have a programmer which understand how the program works in order to delete or edit the data directly in text file when mistake input occur (which is assumed will not happen) and any new car model or assembly section must be also added by the programmer directly in function of generateTable (which is also assumed rarely will happen).

2. Design of the program

2.1 Pseudocode

Pseudocode.txt

```
FUNCTION generateTable()
        ASSIGN  "Menu

                ------------------------------------
                | No.  | Operation                 |
                ------------------------------------
                | 1.   | Parts Creation            |
                | 2.   | Suppliers Creation        |
                | 3.   | Part Update               |
                | 4.   | Part Tracking             |
                | 5.   | Search Function           |
                | -1.  | Quit Program              |
                ------------------------------------
                END Menu

                Part Tracking

                ------------------------------------
                | No. |  Operation                 |
                ------------------------------------
                | 1.  |  Available Quantity        |
                | 2.  |  Short Quantity            |
                | 3.  |  Used Quantity             |
                | -1. |  Quit Part Tracking        |
```

END Part Tracking

Search

| No. | Operation |

| 1. | Part Record |

| 2. | Supplier Detail |

| 3. | Supplier Supplied |

| -1. | Quit Search |

END Search

WBS WAY WBR

Warehouse Code

| Model | Code |

| Bios | WBS |

| Ambry | WAY |

| Barrier | WBR |

END Warehouse Code

ES BWS AS

Assembly Section

```
                    ------------------------------------
                    | Section              | Code |
                    ------------------------------------
                    | Engine Section       | ES   |
                    | Body Work Section    | BWS  |
                    | Air-con Section      | AS   |
                    ------------------------------------
                    END Assembly Section" as data
        Open table.txt file in Write Mode as fileHandler
        Write data into to fileHandler
        Close fileHandler
        Return
ENDFUNCTION

FUNCTION displayTable(tableName)
        Open table.txt file in Read Mode as fileHandler
        READ content from fileHandler into data
        CONVERT data AS ARRAY using NEWLINE as a delimiter
        Close fileHandler
        LOOP count FROM 0 TO Length of data STEP 1
                IF data[count] is equal to tableName
                        ASSIGN result from count + 1 as initialRange
                ELSE
                        IF data[count] is equal to result of "END" + tableName
                                ASSIGN count as endRange
                                Break the loop
                        ENDIF
                ENDIF
                Next count
        ENDLOOP
```

```
    LOOP count FROM initialRange TO endRange STEP 1
        Print data[count]
        Next count
    ENDLOOP
    Return
ENDFUNCTION

FUNCTION verticalDataExtrator(fileName, index, convert)
    DECLARE dataList as empty ARRAY
    DECLARE removeCharacter as ARRAY that include character of 'P' and 'S'
    Open fileName file in Read Mode as fileHandler
    READ content from fileHandler in lines into data
    Close fileHandler
    LOOP items in data
        REMOVE trailing spaces FROM items
        CONVERT items AS ARRAY using TAB as a delimiter
        IF convert is equal to "true" THEN
            LOOP character in removeCharacter
                Replace character in items[index] to ''
                next character
            ENDLOOP
            CONVERT String in items[index] to Integer
        ENDIF
        Append items[index] into dataList
        Next items
    ENDLOOP
    Return dataList
ENDFUNCTION

FUNCTION horizontalDataExtrator(fileName, target, index, condition, loop)
```

```
    DECLARE targetedDataList as empty ARRAY
    Open fileName.txt file in Read Mode as fileHandler
    READ content from fileHandler in lines into data
    Close fileHandler
    LOOP items in data
        REMOVE trailing spaces FROM items
        CONVERT items AS ARRAY using TAB as a delimiter
        IF condition is equal to "==" THEN
            ASSIGN result of Check whether items[index] is equal to
            target as result
        ELSE
            IF condition is equal to '<' THEN
                ASSIGN result of Check whether items[index] is less
                than target as result
            ENDIF
        ENDIF
        IF result is equal to True THEN
            ASSIGN items as targetedData
            IF loop is not equal to "true" THEN
                Return targetedData
            ENDIF
            Append targetedData into targetedDataList
        ENDIF
        Next items
    ENDLOOP
    Return targetedDataList
ENDFUNCTION

FUNCTION record(fileName, data, mode)
    Open fileName file in mode Mode as fileHandler
    LOOP items in data
```

```
            LOOP item in items
                Write item to fileHandler
                Write TAB to fileHandler
                Next item
            ENDLOOP
            Write NEWLINE to fileHandler
            Next items
        ENDLOOP
        Close fileHandler
        Return
ENDFUNCTION

FUNCTION rearrange(fileName)
        DECLARE newData as empty ARRAY
        DECLARE removeCharacter as ARRAY that include character of 'P' and 'S'
        ASSIGN dataList from Call verticalDataExtrator(fileName, 0, "true") as idList
        Sort item in idList with ascending
        ASSIGN idList as rrangedIdList
        Open fileName file in Read Mode as fileHandler
        READ content from fileHandler in lines into data
        Close fileHandler
        LOOP item in arrangedIdList
            LOOP items in data
                REMOVE trailing spaces FROM items
                CONVERT items AS ARRAY using TAB as a delimiter
                LOOP character in removeCharacter
                    Replace character in items[0] to ''
                    IF items[0] can CONVERT to Integer THEN
                        IF items[0] is equal to item THEN
                            ASSIGN result of character + items[0] as
                            items[0]
```

```
                                        Append items into newData
                                        Break the loop
                                    ENDIF
                                ELSE
                                    Pass and Do Nothing
                                ENDIF
                                Next character
                            ENDLOOP
                            Next items
                        ENDLOOP
                        Next item
                    ENDLOOP
                    Call record(fileName, newData, 'w')
                    Return
ENDFUNCTION

FUNCTION idExistanceChecker(fileName, idNo, creationRequirement)
    IF fileName file is exist THEN
        ASSIGN dataList from Call verticalDataExtrator(fileName, 0, "false")
        as idList
        LOOP count FROM 0 TO Length of idList STEP 1
            IF idNo is equal to idList[count] THEN
                Return "exist"
            ENDIF
            Next count
        ENDLOOP
    ENDIF
    ELSE
        Pass and Do Nothing
    IF creationRequirement is equal to "true" THEN
        IF fileName is equal to "parts.txt" THEN
```

```
                Call partsCreation(idNo)
            ELSE
                Call suppliersCreation(idNo)
            ENDIF
        ENDIF
        Return
ENDFUNCTION

FUNCTION codeExistanceChecker(tableName, code)
        Open table.txt file in Read Mode as fileHandler
        READ content from fileHandler into data
        CONVERT data AS ARRAY using NEWLINE as a delimiter
        Close fileHandler
        LOOP count FROM 0 TO Length of data
            IF data[count] is equal to tableName THEN
                ASSIGN data[count - 1] as codeList
                Break the loop
            ENDIF
            next count
        ENDLOOP
        REMOVE trailing spaces FROM codeList
        CONVERT codeList AS ARRAY using TAB as a delimiter
        LOOP count FROM 0 TO Length of codeList
            IF code is equal to codeList[count] THEN
                Return code
            ENDIF
            Next count
        ENDLOOP
        Print "<Invalid Input>"
        Print "Please enter a valid code: "
```

```
        Read code
        CONVERT String in code to Uppercase
        ASSIGN code from Call codeExistanceChecker(tableName, code) as code
        Return code
ENDFUNCTION

FUNCTION quantityAvailability(partId, partQuantity)
        ASSIGN targetedData from Call horizontalDataExtrator("parts.txt", partId, 0,
        "==", "false") as data
        CONVERT String in data[2] to Integer
        IF data[2] is less than partQuantity THEN
                Print "<Quantity exceed available quantity>"
                DOWHILE True
                        IF partQuantity can Convert to Integer THEN
                                Print "How much quantity wanted to use: "
                                Read partQuantity
                                CONVERT String in partQuantity to Integer
                                Break the loop
                        ELSE
                                Pass and Do Nothing
                        ENDIF
                ENDDO
                ASSIGN partQuantity from Call quantityAvailability(partId,
                partQuantity) as partQuantity
        ENDIF
        Return partQuantity
ENDFUNCTION

FUNCTION fileUpdater(fileName, partId, supplierId, partQuantity, operation)
        ASSIGN String from Call idExistanceChecker(partId, fileName, "false") as
        existance
```

```
DECLARE newData as empty ARRAY
IF existance is equal to "exist" THEN
    Open fileName file in Read Mode as fileHandler
    READ content from fileHandler in lines into data
    Close fileHandler
    LOOP items in data
        REMOVE trailing spaces FROM items
        CONVERT items AS ARRAY using TAB as a delimiter
        IF items[0] is equal to partId
            IF operation is equal to '+' THEN
                CONVERT String in items[2] to Integer
                ASSIGN result of items[2] + partQuantity as
                items[2]
            ELSE
                IF operation is equal to '-' THEN
                    CONVERT String in items[2] to Integer
                    ASSIGN result of items[2] -
                    partQuantity as items[2]
                ENDIF
            ENDIF
            CONVERT Integer in items[2] to String
        ENDIF
        Append items into newData
        Next items
    ENDLOOP
    Call record(fileName, newData, 'w')
ELSE
    ASSIGN targetedData from Call horizontalDataExtrator("parts.txt",
    partId, 0, "==", "false") as data
    IF operation is equal to '+' THEN
        ASSIGN 0 + partQuantity as data[2]
```

```
            ELSE
                IF operation is equl to '-' THEN
                    ASSIGN 0 - partQuantity as data[2]
                ENDIF
            ENDIF
            CONVERT Integer in data[2] to String
            IF supplierId is not equal to "none"
                Append supplierId into data
            ENDIF
            Append data into newData
            Call record(fileName, newData, 'a')
            Call rearrange(fileName)
        ENDIF
        Return
ENDFUNCTION

FUNCTION partsCreation(partId)
        IF partId is equal to "none" THEN
            Print "Enter a unique part id(-1 to quit parts creation): "
            Read partId
            ASSIGN "true" as loop
        ELSE
            ASSIGN "false" as loop
        ENDIF
        DOWHILE partId is not equal to "-1"
            DECLARE item as empty ARRAY
            DECLARE items as empty ARRAY
            CONVERT String in partId to Uppercase
            IF Length of partId is not equal to 5 OR partId[0] is not equal to 'P'
            THEN
                Print "<Invalid Id>"
```

Print "<Part Id must start with character 'p' together with 4 number>"

Print "NEWLINE"

Print "Enter a unique part id(-1 to quit parts creation): "

Read partId

Proceed to next iteration

ENDIF

ASSIGN String from Call idExistanceChecker("parts.txt", partId, "false") as existance

IF existance is equal to "exist" THEN

Print "<ID Existed>"

Print "NEWLINE"

Print "Please enter a UNIQUE part id(-1 to quit parts creation): "

Read partId

Proceed to next iteration

ENDIF

Append partId into item

Print "Enter a new part name: "

Read partName

Append partName into item

DOWHILE True

IF partQuantity can CONVERT to Integer THEN

Print "Enter the initial quantity for that part: "

Read partQuantity

CONVERT Integer in partQuantity to String

Break the loop

ELSE

Print "<Invalid Quantity>"

Print NEWLINE

ENDIF

```
ENDDO
Append partQuantity into item
Call displayTable("Warehouse Code")
Print "Please enter the warehouse code: "
Read warehouseCode
CONVERT String in warehouseCode to Uppercase
ASSIGN code from Call codeExistanceChecker("Warehouse Code",
warehouseCode) as warehouseCode
Append warehouseCode into item
Call displayTable("Assembly Section")
Print "Please enter the assembly section code they are used: "
Read assemblySection
CONVERT String in assemblySection to Uppercase
ASSIGN code from Call codeExistanceChecker("Assembly Section",
assemblySection) as assemblySection
Append assemblySection into item
Append item into items
Call record("parts.txt", items, 'a')
Print "Enter a supplier id who supply this part: "
Read supplierId
CONVERT String in supplierId to Uppercase
ASSIGN String from Call idExistanceChecker("suppliers.txt",
supplierId, "false") as existance
IF existance is not equal to "exist" THEN
    Print "<New Id Detected>"
    Print NEWLINE
    Call suppliersCreation(supplierId)
ENDIF
Append supplierid into item
Call record("suppliedParts.txt", items, 'a')
Print "<Part created successfully>"
```

```
            Print NEWLINE
            IF loop is equal to "false" THEN
                Break the loop
            ENDIF
            Print "Enter a unique part id(-1 to quit parts creation): "
            Read partId
        ENDDO
        Call rearrange("parts.txt")
        Call rearrange("suppliedParts.txt")
        Return
ENDFUNCTION

FUNCTION suppliersCreation(supplierId)
        IF supplierId is equal to "none" THEN
            Print "Enter a unique supplier id(-1 to quit suppliers creation): "
            Read supplierId
            ASSIGN "true" as loop
        ELSE
            ASSIGN "false" as loop
        ENDIF
        DOWHILE supplierId is not equal to "-1"
            DECLARE item as empty ARRAY
            DECLARE items as empty ARRAY
            CONVERT String in supplierIf to Uppercase
            IF Length of supplierId is not equal to 5 OR supplierId[0] is not equal
            to 'S' THEN
                Print "<Invalid Id>"
                Print "<Supplier Id must start with character 's' together with 4
                number>"
                Print "NEWLINE"
                Print "Enter a unique supplier id(-1 to quit suppliers creation): "
```

```
        Read supplierId
        Proceed to next iteration
    ENDIF
    ASSIGN String from Call idExistanceChecker("parts.txt", supplierId,
    "false") as existance
    IF existance is equal to "exist" THEN
        Print "<ID Existed>"
        Print "NEWLINE"
        Print "Please enter a UNIQUE supplier id(-1 to quit parts
        creation): "
        Read supplierId
        Proceed to next iteration
    ENDIF
    Append supplierId into item
    Print "Enter a new supplier name: "
    Read supplierName
    Append supplierName into item
    Print "Enter the supplier company address: "
    Read companyAddress
    Append companyAddress into item
    Print "Please enter the supplier phone number: "
    Read phoneNumber
    Append phoneNumber into item
    Append item into items
    Call record("suppliers.txt", items, 'a')
    Print "<Supplier created successfully>"
    Print NEWLINE
    IF loop is equal to "false" THEN
        Break the loop
    ENDIF
    Print "Enter a unique supplier id(-1 to quit suppliers creation): "
```

```
            Read supplierId
        ENDDO
        Call rearrange("suppliers.txt")
        Return
ENDFUNCTION

FUNCTION partUpdate()
        Print "Enter a part id to update(-1 to quit part update): "
        Read partId
        DOWHILE partId is not equal to "-1"
            CONVERT String in partId to Uppercase
            IF Length of partId is not equal to 5 OR partId[0] is not equal to 'P'
            THEN
                Print "<Invalid Id>"
                Print "<Part Id must start with character 'p' together with 4
                number>"
                Print "NEWLINE"
                Print "Enter a part id to update(-1 to quit part update): "
                Read partId
                Proceed to next iteration
            ENDIF
            Call idExistanceChecker("parts.txt", partId, "true")
            Print "Enter any key to supply the part(-1 to use in assembly section): "
            Read decision
            IF decision is equal to "-1" THEN
                Open parts.txt file in Read Mode as fileHandler
                READ content from fileHandler in lines into data
                Close fileHandler
                LOOP items in data
                        REMOVE trailing spaces FROM items
```

```
                CONVERT items AS ARRAY using TAB as a
                delimiter
                IF items[0] is equal to partId THEN
                    Print "Available quantity:", items[2]
                    Break the loop
                ENDIF
                Next items
            ENDLOOP
            DOWHILE True
                IF partQuantity can Convert to Integer THEN
                    Print "How many quantity wanted to use: "
                    Read partQuantity
                    CONVERT String in partQuantity to Integer
                    Break the loop
                ELSE
                    Pass and Do Nothing
                ENDIF
            ENDDO
            ASSIGN result of 0 - partQuantity from Call
            quantityAvailability(partId, partQuantity) as partQuantity
            fileUpdater("usedPart.txt", partId, "none", partQuantity, '-')
        ELSE
            DOWHILE True:
                IF partQuantity can Convert to Integer THEN
                    Print "How many quantity supplied: "
                    Read partQuantity
                    CONVERT String partQuantity to Integer
                    Break the loop
                ELSE
                    Pass and Do Nothing
                ENDIF
```

```
                    ENDDO
                    fileUpdater("suppliedParts.txt", partId, “none” , partQuantity, '+')
                ENDIF
                fileUpdater("parts.txt", partId, "none", partQuantity, '+')
                Print "<Parts update successfully>"
                Print NEWLINE
                Print "Enter a part id to update(-1 to quit part update): "
                Read partId
        Return
ENDFUNCTION

FUNCTION partTracking()
        IF parts.txt file is exist THEN
                Call displayTable("Part Tracking")
                Print "Please choose the operation: "
                Read operation
                DOWHILE operation is not equal to "-1"
                        IF operation is equal to '1' THEN
                                Open parts.txt file in Read Mode as fileHandler
                                READ content from fileHandler in lines into data
                                Close fileHandler
                                LOOP items in data
                                        REMOVE trailing spaces FROM items
                                        CONVERT items AS ARRAY using TAB as a delimiter
                                        Print "ID              :", items[0]
                                        Print "Name            :", items[1]
                                        Print "Quantity Available:", items[2]
                                        Print "Warehouse Code   :", items[3]
                                        Print "Assembly Section :", items[4]
```

```
                Print NEWLINE
                Next items
            ENDLOOP
        ELSE
            IF operation is equal to '2' THEN
                ASSIGN targetedDataList from Call
                horizontalDataExtrator("parts.txt", "10", 2, '<',
                "true") as data
                IF data is an empty ARRAY THEN
                    Print "<No short quantity yet>"
                    Call displayTable("Part Tracking")
                    Print "Please choose the operation: "
                    Read operation
                    Proceed to next iteration
                ENDIF
                Call displayTable("Warehouse Code")
                Print "Enter a warehouse code(-1 to quit short
                quantity tracking): "
                Read warehouseCode
                DOWHILE warehouseCode is not equal to "-1"
                    CONVERT String in warehouseCode to
                    Uppercase
                    ASSIGN code from Call
                    codeExistanceChecker("Warehouse
                    Code", warehouseCode) as
                    warehouseCode
                    DECLARE newData as empty ARRAY
                    LOOP items in data
                        IF items[3] is equal to
                        warehouseCode THEN
                            Append items into
                            newData
                            Print "ID          :",
                            items[0]
```

```
                    Print "Name          :", items[1]
                    Print "Quantity Available:", items[2]
                    Print "Assembly Section :", items[4]
                    Print NEWLINE
                ENDIF
                Next items
            ENDLOOP
            IF newData is an empty ARRAY THEN
                Print "<No short quantity in this warehouse>"
            ENDIF
            Call displayTable("Warehouse Code")
            Print "Enter a warehouse code(-1 to quit short quantity tracking): "
            Read warehouseCode
        ENDDO
    ELSE
        IF operation is equal to '3' THEN
            IF usedParts.txt file is exist THEN
                Open usedParts.txt file in Read Mode as fileHandler
                READ content from fileHandler in lines into data
                Close fileHandler
                Call displayTable("Warehouse Code")
                Print "Enter a warehouseCode(-1 to quit used quantity tracking): "
                Read warehouseCode
                DOWHILE warehouseCode is not equal to "-1"
```

CONVERT String in warehouseCode to Uppercase

ASSIGN code from Call codeExistanceChecker(" Warehouse Code", warehouseCode) as warehouseCode

DECLARE newData as empty ARRAY

LOOP items in data

REMOVE trailing spaces FROM items

CONVERT items AS ARRAY using TAB as a delimiter

IF items[3] is equal to warehouseCode

Append items into newData

Print "ID :", items[0]

Print "Name :", items[1]

Print "Quantity Used :", items[2]

Print "Assembly Section :", items[4]

Print NEWLINE

```
                    ENDIF
                    Next items
                ENDLOOP
                IF newData is an empty ARRAY THEN
                    Print "<No used parts in this warehouse>"
                ENDIF
                Call displayTable("Warehouse Code")
                Print "Enter a warehouseCode(-1 to quit used quantity tracking): "
                Read warehouseCode
            ENDDO
        ELSE
            Print "<No used parts yet>"
        ENDIF
    ELSE
        Print "Invalid Input"
    ENDIF
ENDIF
ENDIF
Call displayTable("Part Tracking")
Print "Please choose the operation: "
```

```
                Read operation
            ENDDO
        ELSE
            Print "<No part recorded yet>"
        ENDIF
        Return
ENDFUNCTION


FUNCTION search()
        IF parts.txt file is exist THEN
            Call displayTable("Search")
            Print "Please choose the operation: "
            Read operation
            DOWHILE operation is not equal to "-1"
                IF operation is equal to '1' THEN
                    Print "Enter part ID you want to search(-1 to quit
                    current search): "
                    Read partId
                    DOWHILE partId is not equal to "-1"
                        CONVERT String in partId to Uppercase
                        DECLARE newData as empty ARRAY
                        IF partId is exist inside parts.txt and
                        suppliedParts.txt file THEN
                            ASSIGN targetedData from Call
                            horizontalDataExtrator("parts.txt",
                            partId, 0, "==", "false") as
                            dataQuantityAvailable
                            LOOP count FROM 0 TO 2 STEP 1
                                Append
                                dataQuantityAvailable[count]
                                into newData
                                Next count
                            ENDLOOP
```

```
ASSIGN targetedData from Call horizontalDataExtrator("suppliedParts.txt", partId, 0, "==", "false") as dataQuantitySupplied

Append dataQuantitySupplied[2] into newData

IF usedParts.txt file is exist THEN

    ASSIGN targetedData from Call horizontalDataExtrator("usedParts.txt", partId, 0, "==", "false") as dataQuantityUsed

    Append dataQuantityUsed[2] into newData

ELSE

    Append 0 into newData

ENDIF

LOOP count FROM 3 TO 4 STEP 1

    Append dataQuantityAvailable[count] into newData

    Next count

ENDLOOP

Print "ID              :", newData[0]

Print "Name            :", newData[1]

Print "Quantity Available:", newData[2]

Print "Quantity Supplied :", newData[3]

IF newData[4] is not equal to 0 THEN

    Print "Quantity Used    :", newData[4]

ENDIF

Print "Warehouse Code   :", newData[5]

Print "Assembly Section :", newData[6]

Print NEWLINE

ELSE
```

```
                    Print "<Part id does not exist>"
                    Print NEWLINE
                ENDIF
                Print "Enter part ID you want to search(-1 to
                quit current search): "
                Read partId
            ENDDO
        ELSE
            IF operation is equal to '2' THEN
                Print "Enter part ID you want to search(-1 to
                quit current search): "
                Read partId
                DOWHILE partId is not equal to "-1"
                    CONVERT String in partId to Uppercase
                    Open suppliedParts.txt file in Read
                    Mode as fileHandler
                    READ content from fileHandler in lines
                    into data
                    Close fileHandler
                    ASSIGN "not exist" as existance
                    LOOP items in data
                        REMOVE trailing spaces FROM
                        items
                        CONVERT items AS ARRAY
                        using TAB as a delimiter
                        IF items[0] is not equal to partId
                            Proceed to next iteration
                        ENDIF
                        ASSIGN targetedData from Call
                        horizontalDataExtrator("suppliers
                        .txt", items[5], 0, "==", "false")
                        as items
                        Print "ID          :", items[0]
                        Print "Name        :", items[1]
```

```
                    Print "Company Address :", items[2]
                    Print "Phone Number    :", items[3]
                    Print NEWLINE
                    ASSIGN "exist" as existance
                    Break the loop
                    Next items
                ENDLOOP
                IF existance is equal to "not exist" THEN
                    Print "<Part id does not exist>"
                    Print NEWLINE
                ENDIF
                Print "Enter part ID you want to search(-1 to quit current search): "
                Read partId
            ENDDO
        ELSE
            IF operation is equal to '3' THEN
                Print "Enter supplier ID you want to search(-1 to quit current search): "
                Read supplierId
                DOWHILE supplierId is not equal to "-1"
                    ASSIGN targetedDataList from Call horizontalDataExtrator("suppliedParts.txt", supplierId, 5, "==", "true") as data
                    IF data is an empty ARRAY THEN
                        Print "<Supplier id does not exist OR this supplier
```

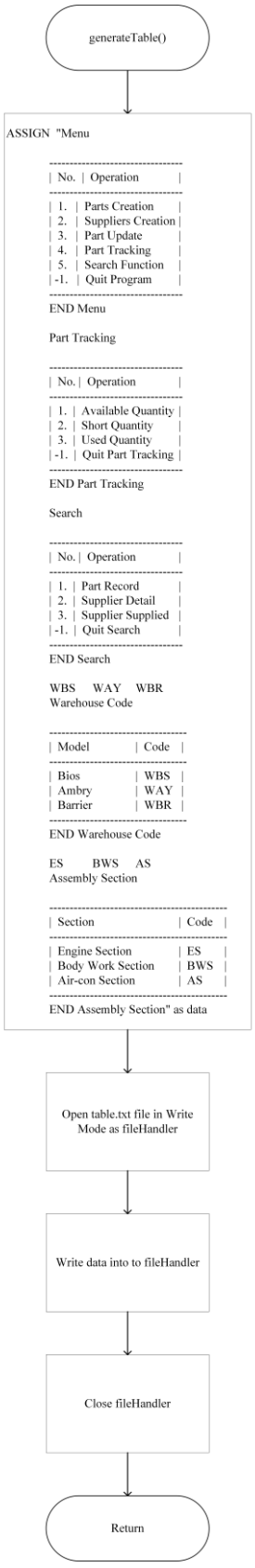
```
                    haven't supply any part
                    yet>"
                    Print NEWLINE
                    Print "Enter supplier ID
                    you want to search(-1 to
                    quit current search): "
                    Read supplierId
                    Proceed to next iteration
                ENDIF
                LOOP items in data
                    Print "ID          :",
                    items[0]
                    Print "Name        :",
                    items[1]
                    Print "Quantity Supplied
                    :", items[2]
                    Print "Warehouse Code
                    :", items[3]
                    Print "Assembly Section
                    :", items[4]
                    Print NEWLINE
                    Next items
                ENDLOOP
                Print "Enter supplier ID you want
                to search(-1 to quit current
                search): "
                Read supplierId
            ENDDO
        ELSE
            Print "Invalid Input"
        ENDIF
    ENDIF
ENDIF
Call displayTable("Search")
```

```
                Print "Please choose the operation: "
                Read operation
            ENDDO
        ELSE
            Print "<No part recorded yet>"
        ENDIF
        Return
ENDFUNCTION

FUNCTION menu()
        Call displayTable("Menu")
        Print "Please choose the operation: "
        Read operation
        DOWHILE operation is not equal to "-1"
            IF operation is equal to '1' THEN
                Call partsCreation("none")
            ELSE
                IF operation is equal to '2' THEN
                    Call suppliersCreation("none")
                ELSE
                    IF operation is equal to '3' THEN
                        Call partUpdate()
                    ELSE
                        IF operation is equal to '4' THEN
                            Call partTracking()
                        ELSE
                            IF operation is equal to '5' THEN
                                Call search()
                            ELSE
                                Print "<Invalid Input>"
```

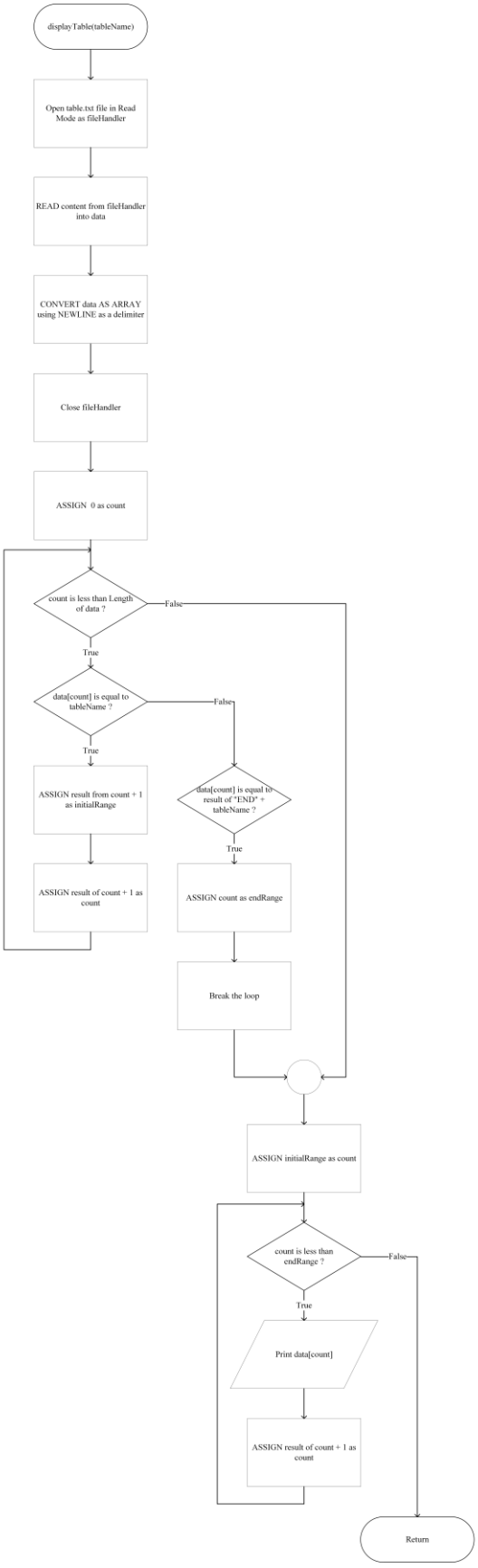
```
                                        ENDIF
                                    ENDIF
                                ENDIF
                            ENDIF
                        ENDIF
                        Call displayTable("Menu")
                        Print "Please choose the operation: "
                        Read operation
                ENDDO
                Return
ENDFUNCTION

PROGRAM Automobile Parts Inventory Management System
BEGIN
        generateTable()
        menu()
END
```

2.2 Flowchart

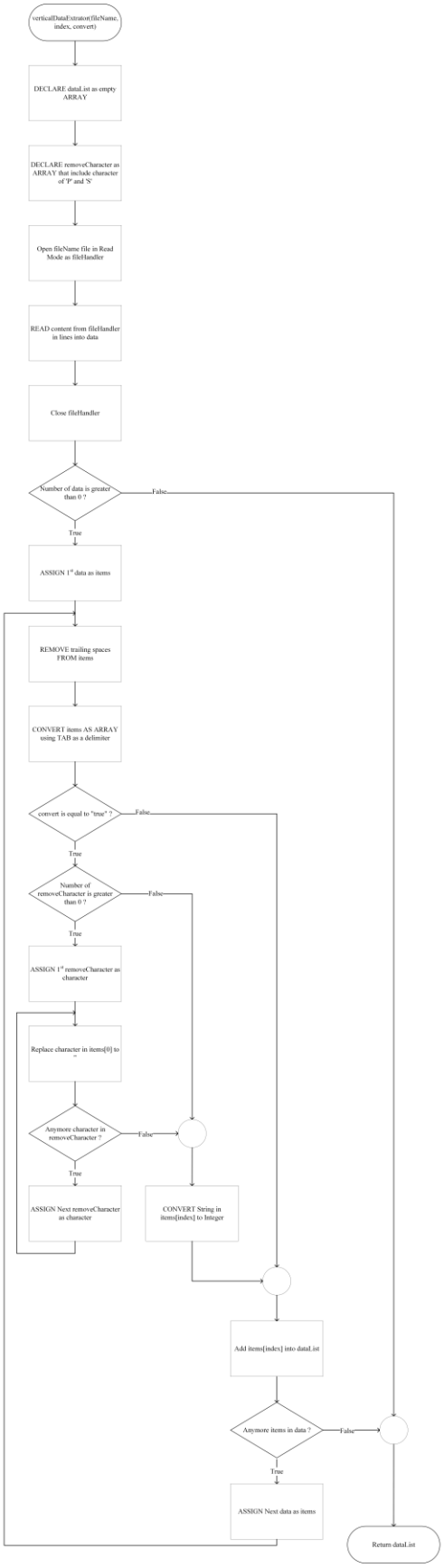
2.2.1 generateTable Function



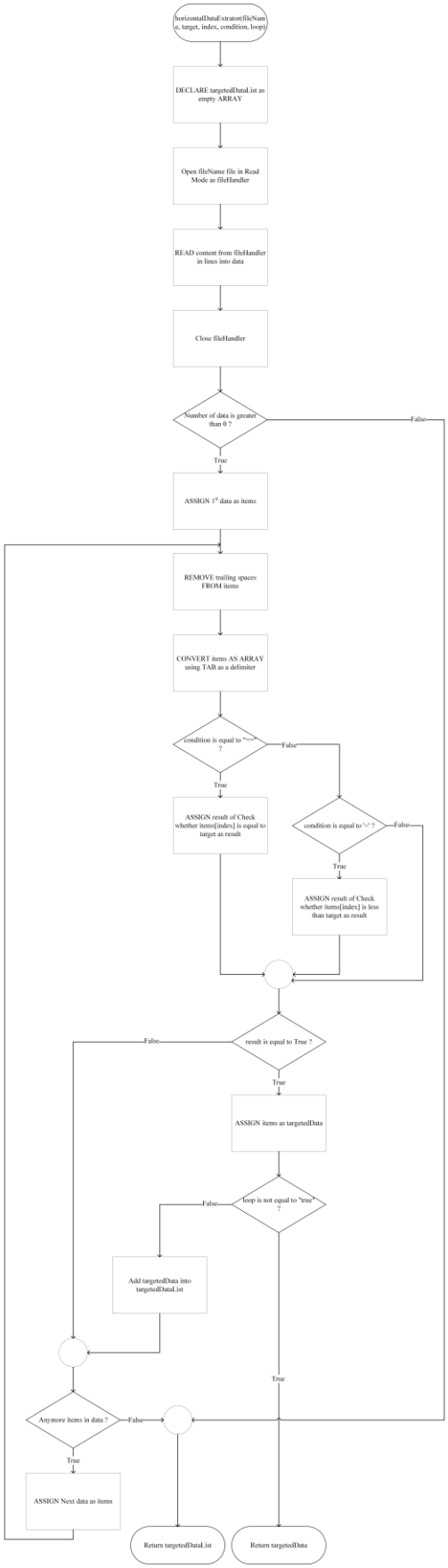
2.2.2 displayTable Function



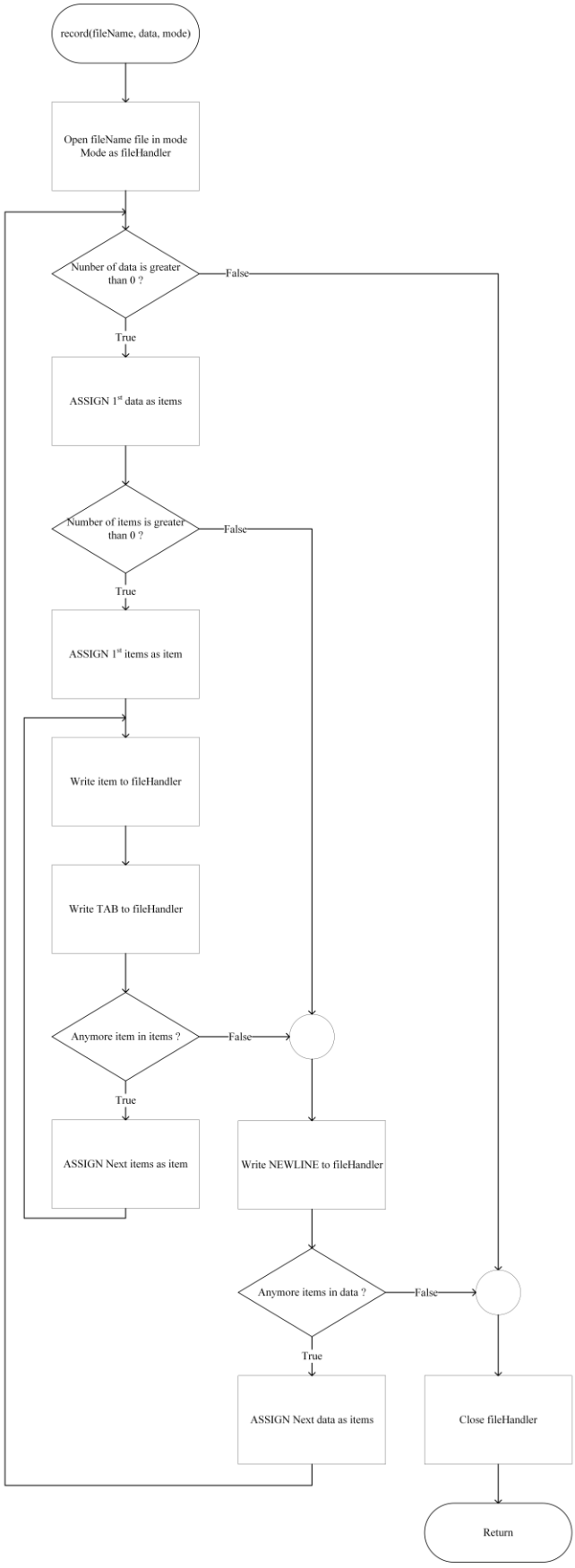
2.2.3 verticalDataExtractor Function



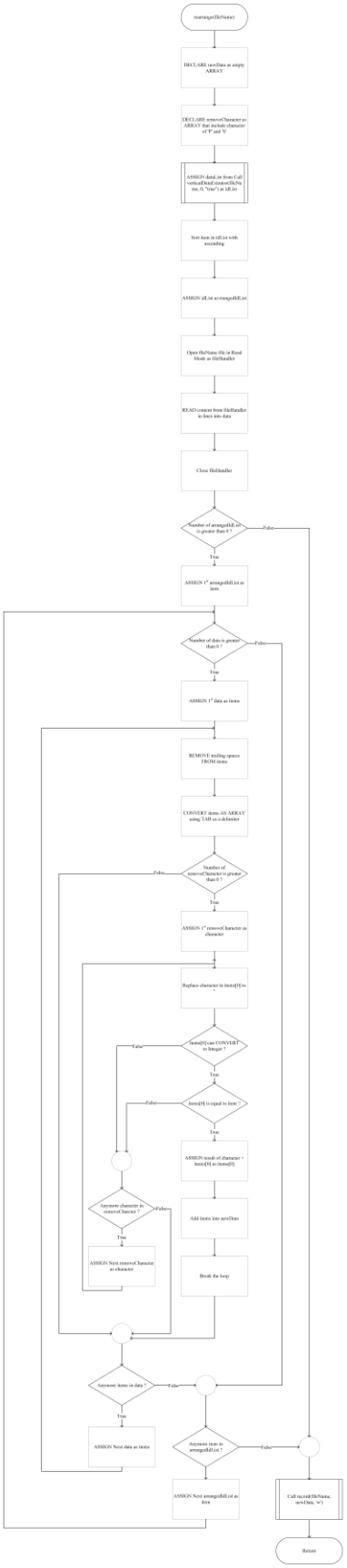
2.2.4 horizontalDataExtractor Function



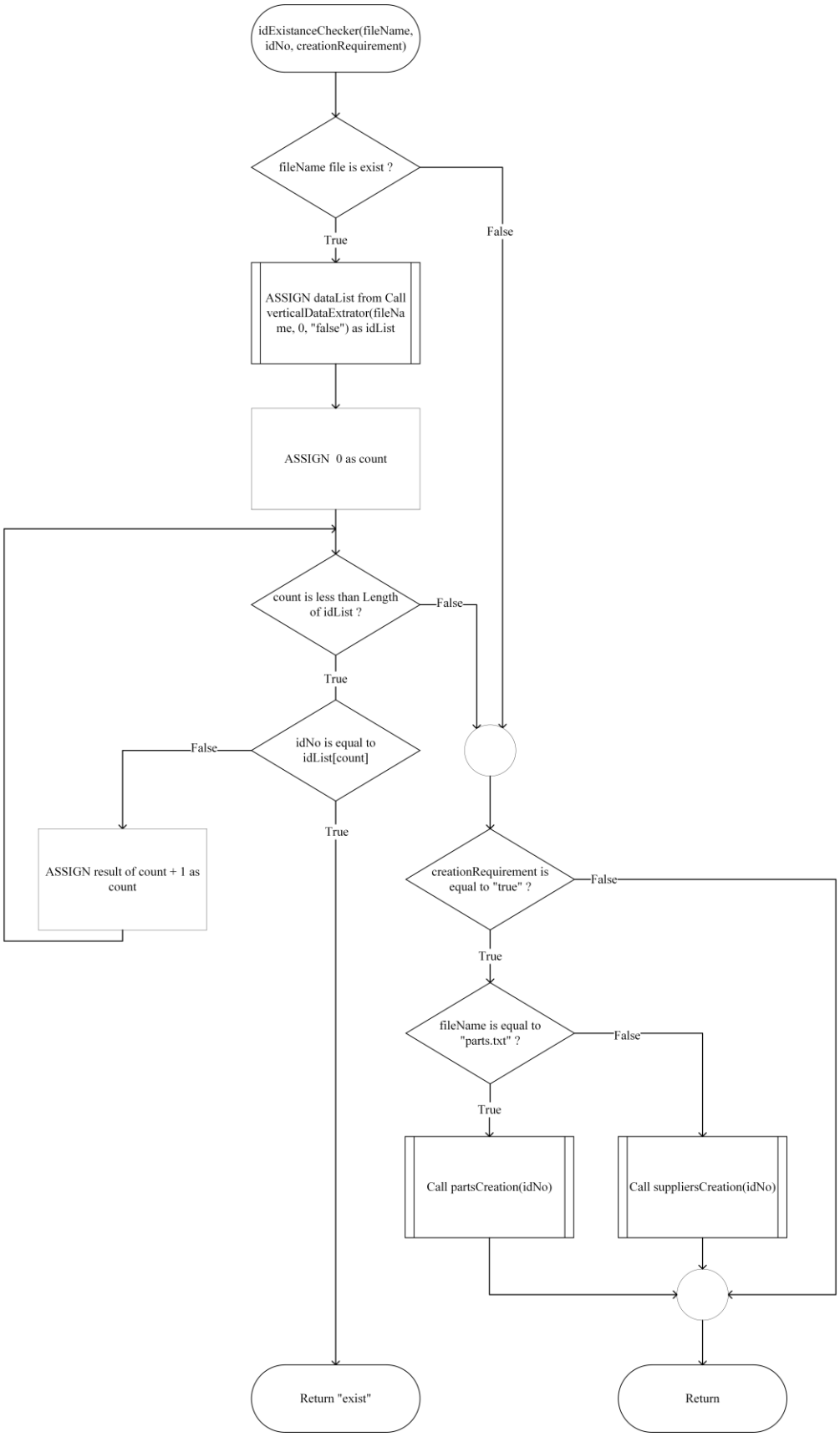
2.2.5 record Function



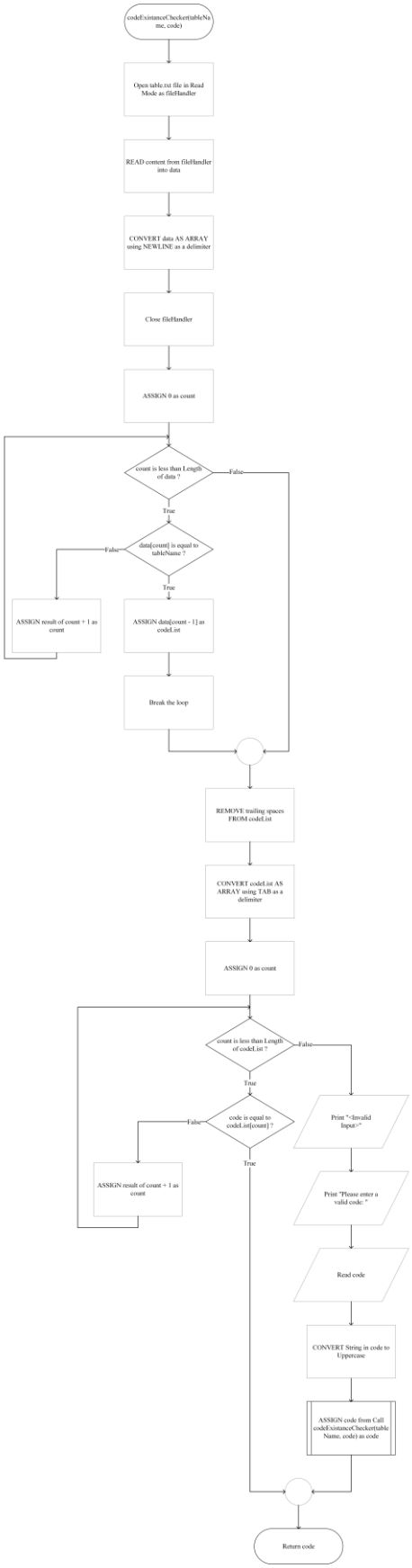
2.2.6 rearrange Function



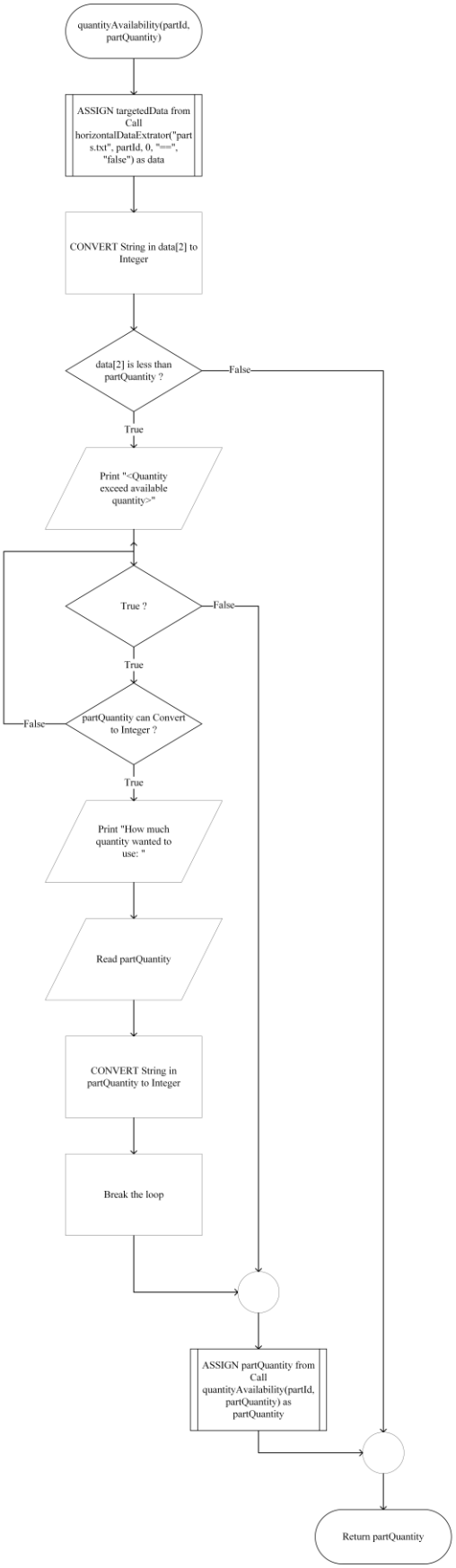
2.2.7 idExistanceChecker Function



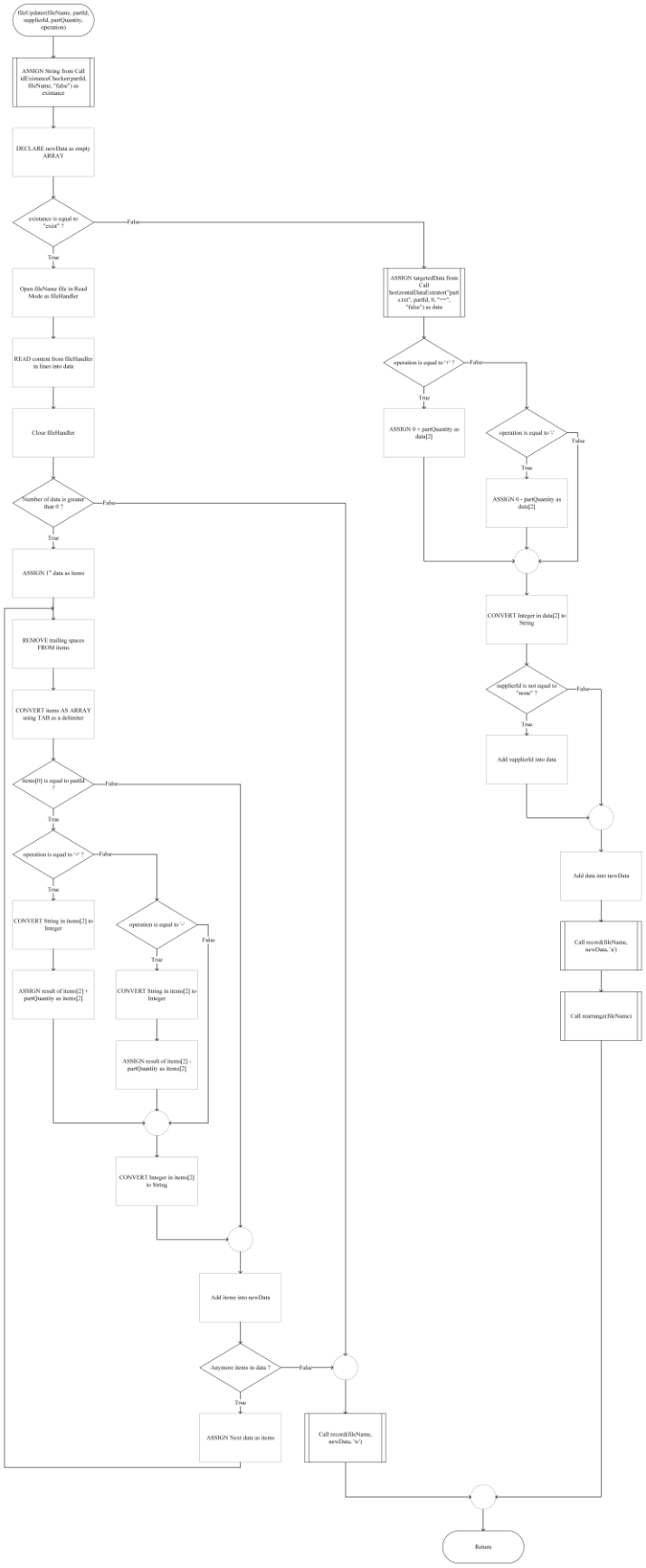
2.2.8 codeExistanceChecker Function



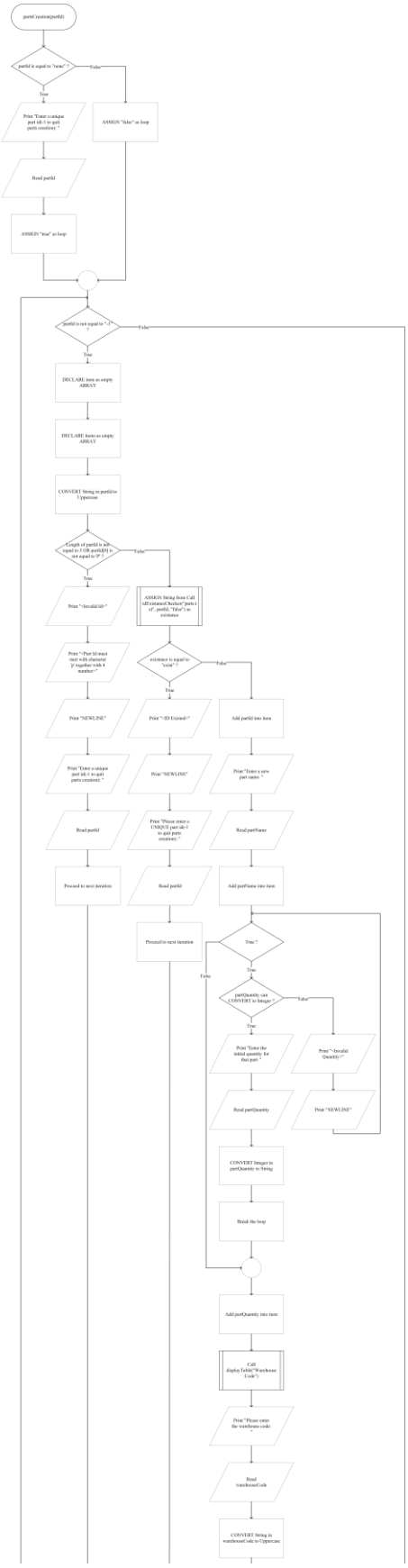
2.2.9 quantityAvailability Function

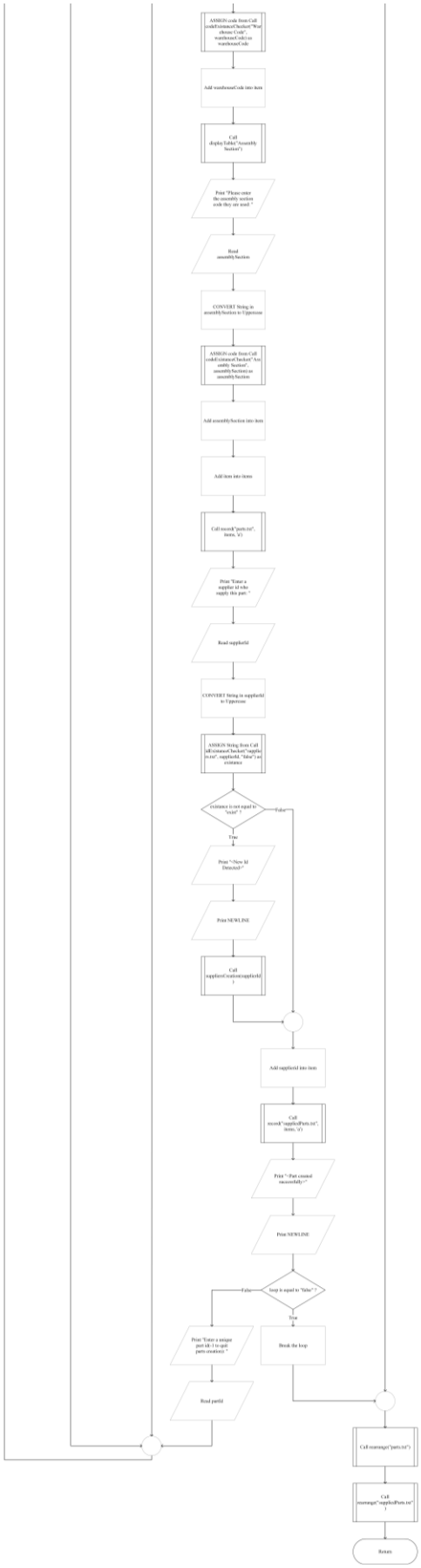


2.2.10 fileUpdater Function

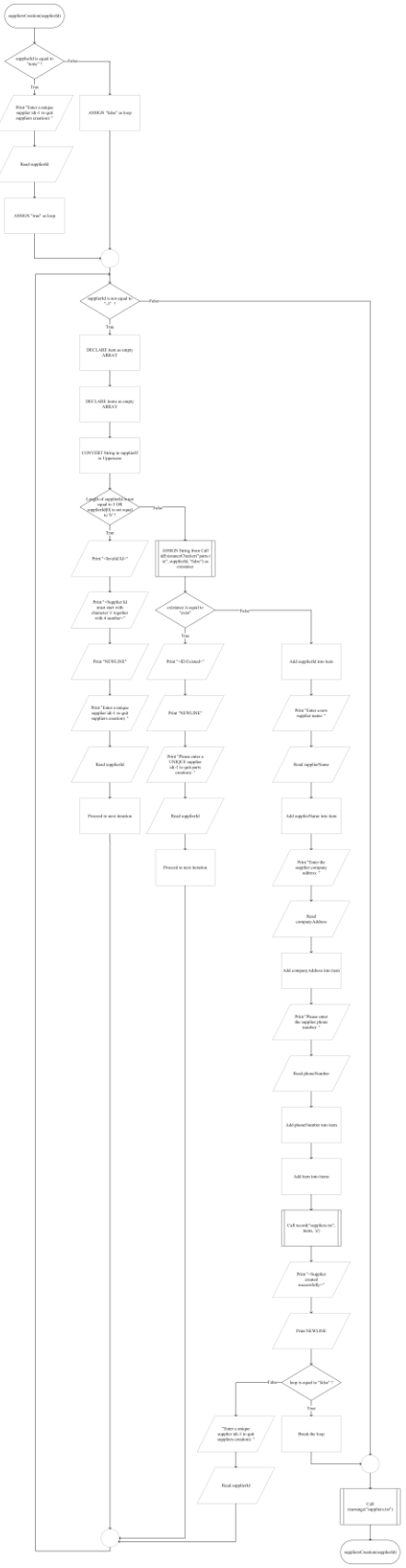


2.2.11 partsCreation Function

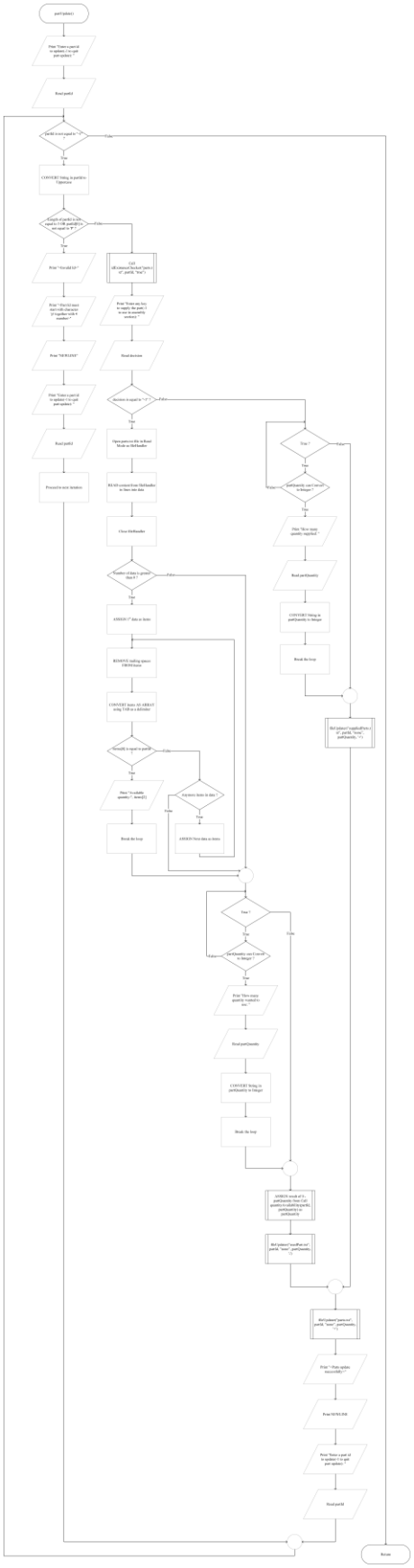



ASSIGN code from Call codeExistenceChecker("Warehouse Code", warehouseCode) to warehouseCode
Add warehouseCode into item
Call displayTable("Assembly Section")
Print "Please enter the assembly section code they are used."
Read assemblySection
CONVERT String in assemblySection to Uppercase
ASSIGN code from Call codeExistenceChecker("Assembly Section", assemblySection) to assemblySection
Add assemblySection into item
Add item into items
Call recordA("parts.txt", items, 'e')
Print "Enter a supplier id who supply this part."
Read supplierId
CONVERT String in supplierId to Uppercase
ASSIGN String from Call idExistenceChecker("supplies.txt", supplierId, "false") as existence
existence is not equal to "exist" ?
True
False
Print "<New Id Detected>"
Print NEWLINE
Call supplierCreation(supplierId)
Add supplierId into item
Call recordA("suppliedParts.txt", items, 'e')
Print "<Part created successfully>"
Print NEWLINE
loop is equal to "false" ?
True
False
Print "Enter a unique part id (-1 to quit parts creation)."
Break the loop
Read partId
Call rearrange("parts.txt")
Call rearrange("suppliedParts.txt")
Return

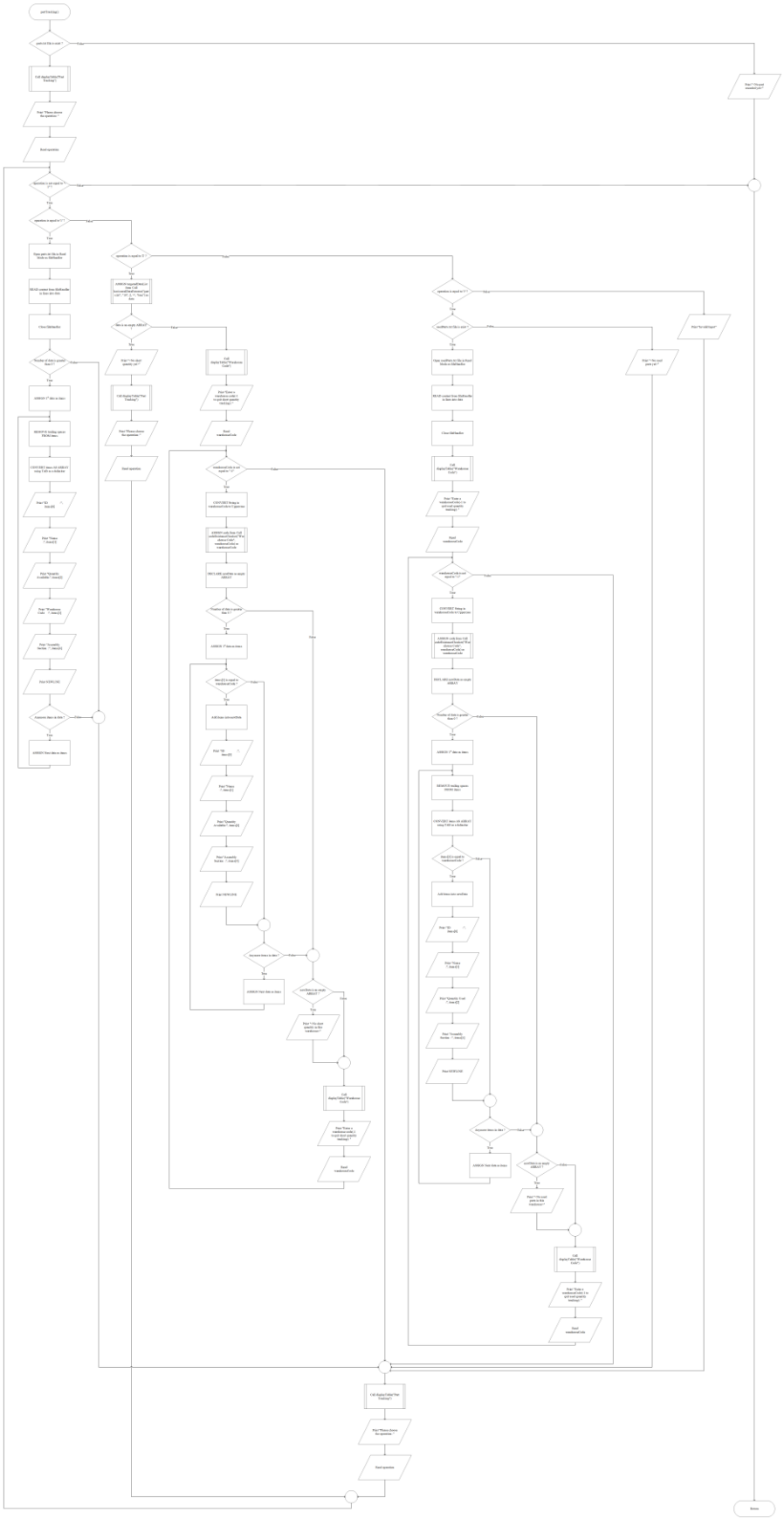
2.2.12 suppliersCreation Function



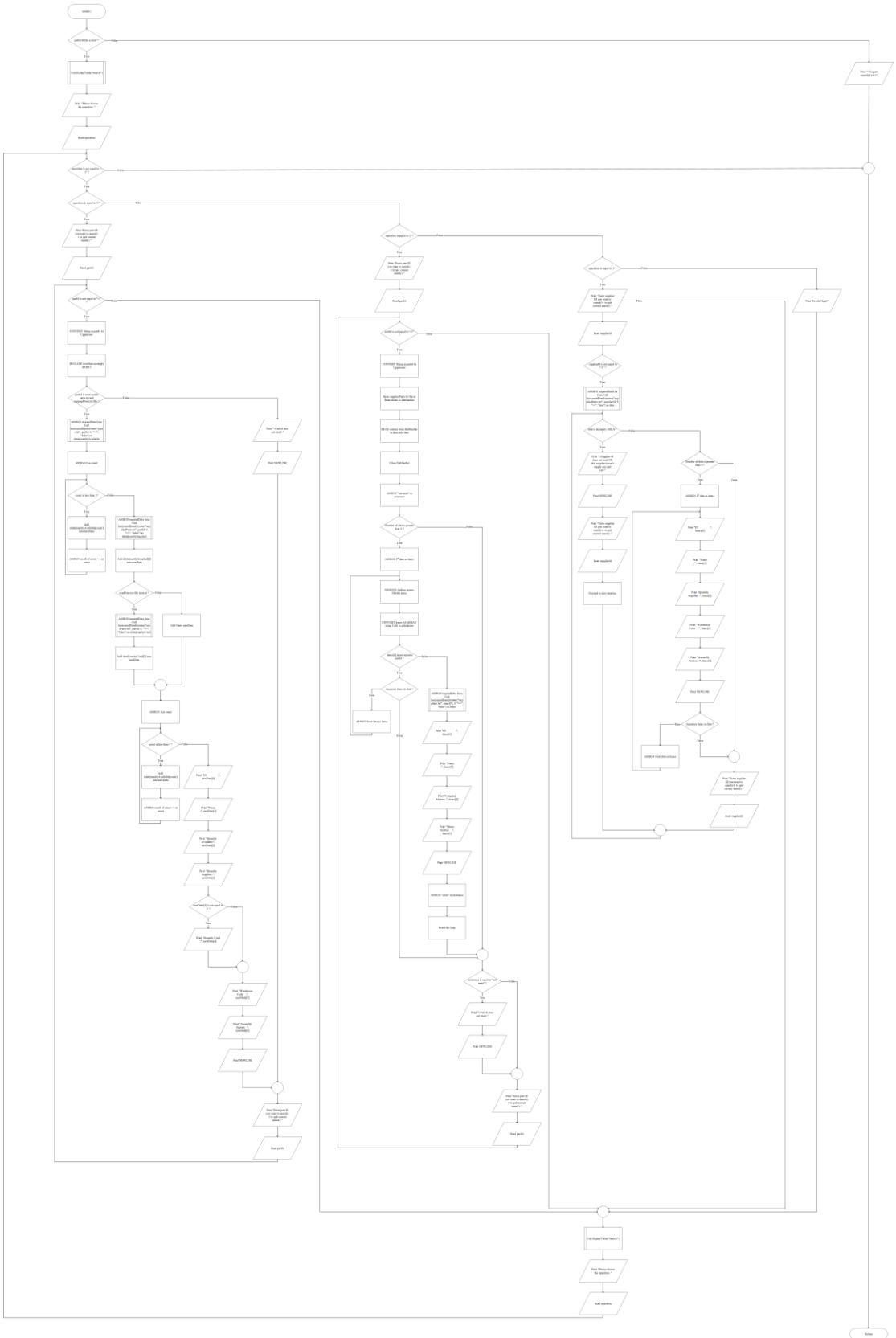
2.2.13 partUpdate Function



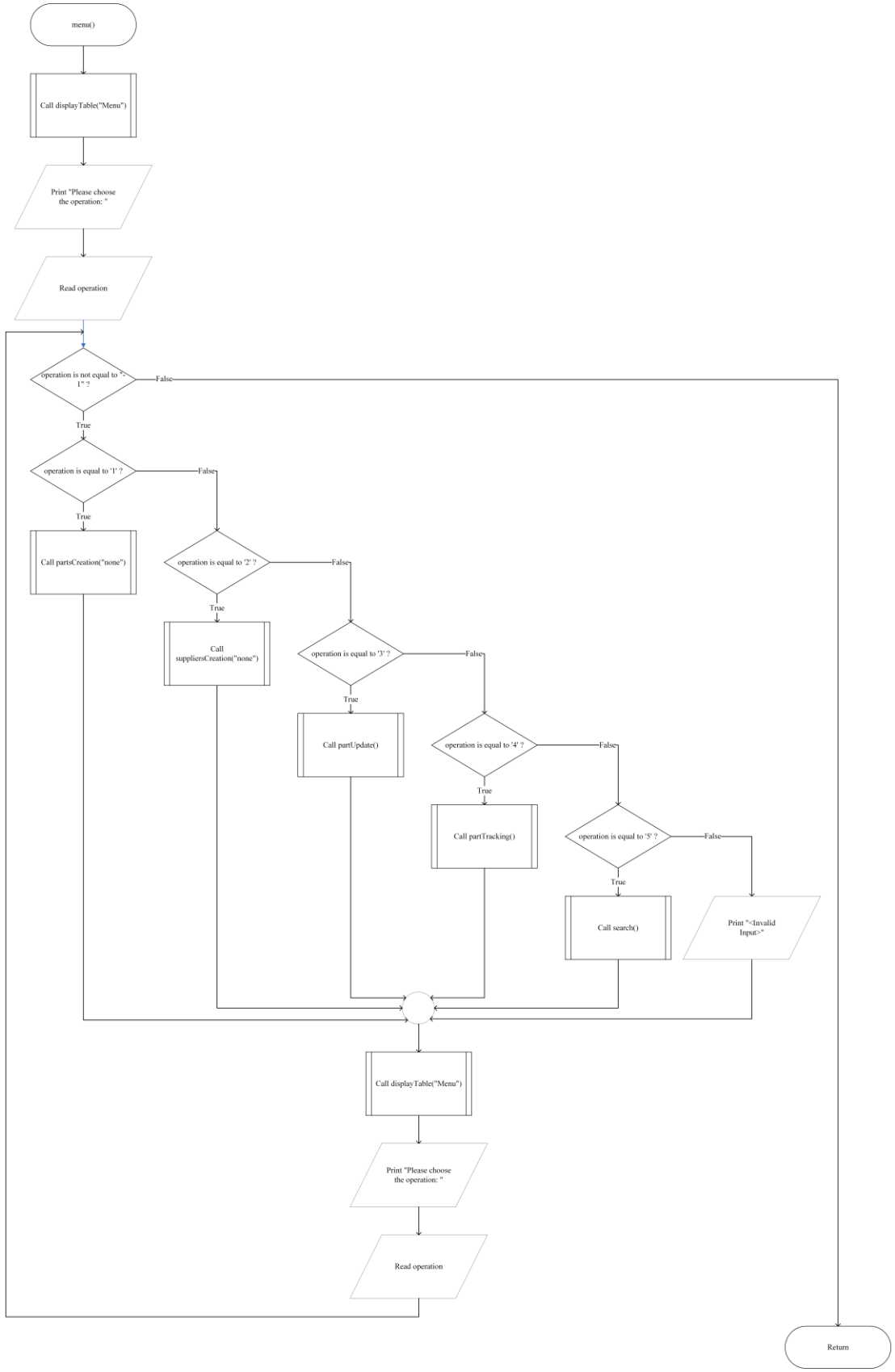
2.2.14 partTracking Function



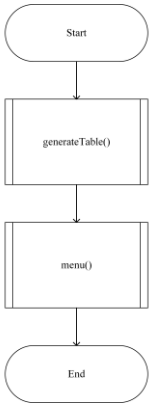
2.2.15 search Function



2.2.16 menu Function



2.2.17 Main Function



3. Source code

```
#Sia DE LONG
#TP060810
def generateTable():
    #Declare all the the table needed into data
    #Any new car model and assembly section will be editing from here
    data = """Menu

-------------------------------------------
| No. | Operation              |
-------------------------------------------
| 1.  | Parts Creation         |
| 2.  | Suppliers Creation     |
| 3.  | Part Update            |
| 4.  | Part Tracking          |
| 5.  | Search Function              |
|-1.  | Quit Program           |
-------------------------------------------
END Menu

Part Tracking

-------------------------------------------
| No.| Operation              |
-------------------------------------------
| 1. | Available Quantity     |
| 2. | Short Quantity         |
| 3. | Used Quantity          |
|-1. | Quit Part Tracking     |
-------------------------------------------
```

END Part Tracking

Search

```
-----------------------------------------
| No. | Operation             |
-----------------------------------------
| 1.  | Part Record           |
| 2.  | Supplier Detail       |
| 3.  | Supplier Supplied     |
| -1. | Quit Search           |
-----------------------------------------
```

END Search

WBS WAY WBR

Warehouse Code

```
-----------------------------------------
| Model              | Code |
-----------------------------------------
| Bios               | WBS  |
| Ambry              | WAY  |
| Barrier            | WBR  |
-----------------------------------------
```

END Warehouse Code

ES BWS AS

Assembly Section

```
-----------------------------------------
```

```
| Section                | Code |
-----------------------------------------
| Engine Section         | ES   |
| Body Work Section      | BWS  |
| Air-con Section        | AS   |
-----------------------------------------
END Assembly Section"""
    #Create table.txt file and write the data into it
    fileHandler = open("table.txt", 'w')
    fileHandler.write(data)
    fileHandler.close()
    return
#Assumed this program is only for Bios, Ambry and Barrier
#Assumed all warehouse will have the same assembly sections
#Assumed there will ahve a programmer who can edit this table to add new car model or assembly section


def displayTable(tableName):
    #Open table.txt file and read it to data then close instantly
    fileHandler = open("table.txt", 'r')
    data = fileHandler.read().split('\n')
    fileHandler.close()
    #Read the data line by line
    for count in range(len(data)):
        if data[count] == tableName:
            #If encounter the desired table name then save the next line in initial range for later
            initialRange = count + 1
        elif data[count] == "END " + tableName:
```

```
            #If encounter END of table name then save the line in endRange for later then
stop the loop afterward
            endRange = count
            break
    #Print out the desired table
    for count in range(initialRange, endRange):
        print(data[count])
    return



def verticalDataExtrator(fileName, index, convert):
    #Declare an array to store related data
    dataList = []
    #Declare an array to delete charater in related data
    removeCharacter = ['P', 'S']
    #Open the desired file and read it as data then close instantly
    fileHandler = open(fileName, 'r')
    data = fileHandler.readlines()
    fileHandler.close()
    #Process the data in smaller list
    for items in data:
        #Abstract out the data
        items = items.rstrip().split('\t')
        #If convert is equal to "true" then replace the character
        #Which exist in removeCharacter with '0'
        #Then convert the data to integer
        if convert == "true":
            for character in removeCharacter:
                items[index] = items[index].replace(character, '0')
            items[index] = int(items[index])
        #Save the small list into a new array
```

```
        dataList.append(items[index])
    #Return the array to function which called it
    return dataList
#Assumed only string that have character 'p' or 's' infront will be proccessed

def horizontalDataExtrator(fileName, target, index, condition, loop):
    #Declare an array to store related data
    targetedDataList = []
    #Open the desired file and read it as data then close instantly
    fileHandler = open(fileName, 'r')
    data = fileHandler.readlines()
    fileHandler.close()
    #Process the data in smaller list
    for items in data:
        items = items.rstrip().split('\t')
        #Abstract out the data
        #Get the result following with the condition
        #The element inside small list will be converted to integer if condition is '<'
        if condition == "==":
            result = items[index] == target
        elif condition == '<':
            result = int(items[index]) < target
        #If the result id True then the small list will become the targetedData
        if result:
            targetedData = items
            #If the loop is not equal to "true" then only the one small list will be return
            if loop != "true":
                return targetedData
            #Save the small list into a new array if loop is "false"
            targetedDataList.append(targetedData)
```

```
    #Return the array to function which called it
    return targetedDataList
#Assumed only condition of "==" and '<' will exists in the program



def record(fileName, data, mode):
    #Open the desired file and either append or rewrite it
    fileHandler = open(fileName, mode)
    #Process the data in smaller list
    for items in data:
        #Process the smaller list in element
        for item in items:
            #Write the element one by one with TAB between them
            fileHandler.write(item)
            fileHandler.write('\t')
        #After all element in a smaller list already write in the file then enter a
NEWLINE
        fileHandler.write('\n')
    #After all process has been done then close the file
    fileHandler.close()
    return



def rearrange(fileName):
    #Declare an array to store related data
    newData = []
    #Declare an array to delete charater in related data
    removeCharacter = ['P', 'S']
    #Get all ids that originally exist in desired file and convert it to integer
    idList = verticalDataExtrator(fileName, 0, "true")
    #Arrange the ids in ascending order then store in a new array
```

```
    arrangedIdList = sorted(idList)
    #Open the desired file and read it as data then close instantly
    fileHandler = open(fileName, 'r')
    data = fileHandler.readlines()
    fileHandler.close()
    #To arrange the data in ascending order of part id
    #Process the arrangedIdList in element
    for item in arrangedIdList:
        #Process the data in smaller list
        for items in data:
            #Abstract out the data
            items = items.rstrip().split('\t')
            #Replace the character in first element of smaller list which exist in
            removeCharacter with '0'
            for character in removeCharacter:
                items[0] = items[0].replace(character, '0')
             #If the character successfully replaced then compared it with element in
            arrangedIdList
                #Else remove the next character
                try:
                    if int(items[0]) == item:
                        #If found the smaller list then put back the character
                        items[0] = character + items[0]
                        #Save the small list into a new array
                        newData.append(items)
                        break
                except ValueError:
                    pass
    #Rewrite the data into desired file to arrange it
    record(fileName, newData, 'w')
    return
```

```
def idExistanceChecker(fileName, idNo, creationRequirement):
    #Try to search the id in desired file
    try:
        #Get all ids that originally exist in desired file without convert
        idList = verticalDataExtrator(fileName, 0, "false")
        for count in range(len(idList)):
            if idNo == idList[count]:
                return "exist"
    #If the file havent created yet then proceed to next if statement
    except FileNotFoundError:
        pass
    #If the function which called it want to proceed to desired creation
    #Then call the creation follow with the idNo
    if creationRequirement == "true":
        if fileName == "parts.txt":
            partsCreation(idNo)
        else:
            suppliersCreation(idNo)
    return
#Assumed only part and supplier creation in the entire program


def codeExistanceChecker(tableName, code):
    #Open table.txt file and read it as data then close instantly
    fileHandler = open("table.txt", 'r')
    data = fileHandler.read().split('\n')
    fileHandler.close()
    #Find out the line which have the code in particular table
```

```
    for count in range(len(data)):
        if data[count] == tableName:
            codeList = data[count - 1]
            break
    #Abstact out the code
    codeList = codeList.rstrip().split('\t')
    #Run throught the code in the line
    #If found the same code then return to the function which called it
    for count in range(len(codeList)):
        if code == codeList[count]:
            return code
    #Else Request the user to input the code again
    print("<Invalid Input>")
    code = input("Please enter a valid code: ")
    code = code.upper()
    code = codeExistanceChecker(tableName, code)
    return code


def quantityAvailability(partId, partQuantity):
    #Get all the detail from parts.txt file which have the same part id as data
    data = horizontalDataExtrator("parts.txt", partId, 0, "==", "false")
    #If the quantity assembly secion requested is exceed available quantity
    #Then request the user to input quantity again
    if int(data[2]) < partQuantity:
        print("<Quantity exceed available quantity>")
        #Loop until get a quantity in integer
        while True:
            try:
                partQuantity = int(input("How much quantity wanted to use: "))
```

```
            break
        except ValueError:
            pass
        #Call again the same function to check the availability and receive the quantity
        as partQuantity
    partQuantity = quantityAvailability(partId, partQuantity)
  #Finally return the valid quantity
  return partQuantity
#Assumed no negative integer input



def fileUpdater(fileName, partId, supplierId, partQuantity, operation):
  #Check either the data with same partId already existed or not
  existance = idExistanceChecker(fileName, partId, "false")
  #Declare an array to store related data
  newData = []
  #If existed then just edit the quantity then rewrite to the file
  if existance == "exist":
    #Open the desired file and read it as data then close instantly
    fileHandler = open(fileName, 'r')
    data = fileHandler.readlines()
    fileHandler.close()
    #Process the data in smaller list
    for items in data:
      items = items.rstrip().split('\t')
      #Compared part id smaller by smaller list
      #Calculate it when found it
      if items[0] == partId:
        if operation == '+':
          items[2] = int(items[2]) + partQuantity
        elif operation == "-":
```

```
                items[2] = int(items[2]) - partQuantity
            #Convert it back to String
            items[2] = str(items[2])
        #Save the small list into a new array
        newData.append(items)
    #Rewrite the content in the file with data in new array
    record(fileName, newData, 'w')
  else:
    #Get data from parts.txt file which have the same id
    data = horizontalDataExtrator("parts.txt", partId, 0, "==", "false")
    #Calculate it
    if operation == '+':
        data[2] = 0 + partQuantity
    elif operation == "-":
        data[2] = 0 - partQuantity
    #Convert it back to String
    data[2] = str(data[2])
    #Add supplierId to the data if needed
    if supplierId != "none":
        data.append(supplierId)
    #Save the data into a new array
    newData.append(data)
    #Append the content in the file with data in new array amd rearrange it
    record(fileName, newData, 'a')
    rearrange(fileName)
  return


def partsCreation(partId):
  #If still doesnt have any part id declared yet then read one
```

```
    if partId == "none":
        partId = input("Enter a unique part id(-1 to quit parts creation): ")
        #Let it allow to loop
        loop = "true"
    else:
        #If the part id already declared then dont let the function to loop
        loop = "false"
    #"-1" as sentinel value
    while partId != "-1":
        #Declare a small list and a master list
        item = []
        items = []
        #Check the validation of part id
        partId = partId.upper()
        if len(partId) != 5 or partId[0] != 'P':
            print("<Invalid Id>\n<Part Id must start with character 'p' together with 4
            number>\n")
            partId = input("Enter a unique part id(-1 to quit suppliers creation): ")
            continue
        existance = idExistanceChecker("parts.txt", partId, "false")
        if existance == "exist":
            print("<ID Existed>\n")
            partId = input("Please enter a UNIQUE part id(-1 to quit parts creation): ")
            continue
        #Store the part id inside small list when it is valid
        item.append(partId)
        #Read part name and store in small list
        partName = input("Enter a new part name: ")
        item.append(partName)
        #Loop until get a quantity in integer
```

```
        while True:
            try:
                partQuantity = int(input("Enter the initial quantity for that part: "))
                #Convert it back to String
                partQuantity = str(partQuantity)
                break
            except ValueError:
                print("<Invalid Quantity>\n")
        #Store quantity, warehouseCode and assemblySection inside small list when it is
valid
        item.append(partQuantity)
        displayTable("Warehouse Code")
        warehouseCode = input("Please enter the warehouse code: ")
        warehouseCode = warehouseCode.upper()
        #Check the validation of warehouse code
        warehouseCode = codeExistanceChecker("Warehouse Code", warehouseCode)
        item.append(warehouseCode)
        displayTable("Assembly Section")
        assemblySection = input("Please enter the assembly section code they are used:
")
        assemblySection = assemblySection.upper()
        #Check the validation of assembly section
        assemblySection = codeExistanceChecker("Assembly Section",
assemblySection)
        item.append(assemblySection)
        #Store the small list to master list then append it to parts.txt file
        #To prevent user input the same id in the same attempt
        items.append(item)
        record("parts.txt", items, 'a')
        #Read supplierId who supplied it
        supplierId = input("Enter a supplier id who supply this part: ")
```

```
        supplierId = supplierId.upper()
        existance = idExistanceChecker("suppliers.txt", supplierId, "false")
        #If new id detected then go to suppliersCreation to fill in the detail
        if existance != "exist":
            print("<New Id Detected>\n")
            suppliersCreation(supplierId)
        item.append(supplierId)
        #Store the id to small list then append the master list to suppliers.txt file
        record("suppliedParts.txt", items, 'a')
        print("<Part created successfully>\n")
        #Stop the loop if loop is "false"
        if loop == "false":
            break
        #Read a new part id then proceed to next iteration
        partId = input("Enter a unique part id(-1 to quit parts creation): ")
    #Rearrange all the file then return
    rearrange("parts.txt")
    rearrange("suppliedParts.txt")
    return
#Assumed no mistake when input details



def suppliersCreation(supplierId):
    #If still doesnt have any supplier id declared yet then read one
    if supplierId == "none":
        supplierId = input("Enter a unique supplier id(-1 to quit suppliers creation): ")
        #Let it allow to loop
        loop = "true"
    else:
        #If the part id already declared then dont let the function to loop
```

```
        loop = "false"
    #"-1" as sentinel value
    while supplierId != "-1":
        #Declare a small list and a master list
        item = []
        items = []
        supplierId = supplierId.upper()
        #Check validation of supplier id
        if len(supplierId) != 5 or supplierId[0] != 'S':
            print("<Invalid Id>\n<Supplier Id must start with character 's' together with 4
            number>\n")
            supplierId = input("Enter a unique supplier id(-1 to quit suppliers creation): ")
            continue
        existance = idExistanceChecker("suppliers.txt", supplierId, "false")
        if existance == "exist":
            print("<ID Existed>")
            supplierId = input("\nPlease enter a UNIQUE supplier id(-1 to quit suppliers
creation):
            ")
            continue
        #Store related details into small list
        item.append(supplierId)
        supplierName = input("Enter new supplier name: ")
        item.append(supplierName)
        companyAddress = input("Enter supplier company address: ")
        item.append(companyAddress)
        phoneNumber = input("Enter supplier phone number: ")
        item.append(phoneNumber)
        #Store the small list into master list
        items.append(item)
        #Append master list to suppliers.txt
```

```
        record("suppliers.txt", items, 'a')
        print("<Supplier created successfully>\n")
        #Stop the loop if loop is "false"
        if loop == "false":
            break
        #Read a new supplier id then proceed to next iteration
        supplierId = input("Enter a unique supplier id(-1 to quit suppliers creation): ")
    #Rearrange the file then return
    rearrange("suppliers.txt")
   return
#Assumed no mistake when input details



def partUpdate():
    #Read a part id first
    partId = input("Enter a part id to update(-1 to quit part update): ")
    #"-1" as sentinel value
    while partId != "-1":
        partId = partId.upper()
        #Check validation and existance
        if len(partId) != 5 or partId[0] != 'P':
            print("<Invalid Id>\n<Part Id must start with character 'p' together with 4
            number>\n")
            partId = input("Enter a part id to update(-1 to quit part update): ")
            continue
        #If doesnt then create it in part creation first
        idExistanceChecker("parts.txt", partId, "true")
        #Let the user choose either to increase or decrease quantity
        decision = input("Enter any key to supply the part(-1 to use in assembly section):
")
        #Process of using part
```

```
    if decision == "-1":
        fileHandler = open("parts.txt", 'r')
        data = fileHandler.readlines()
        fileHandler.close()
        for items in data:
            items = items.rstrip().split('\t')
            if items[0] == partId:
                print("Available quantity:", items[2])
                break
        while True:
            try:
                partQuantity = int(input("How many quantity wanted to use: "))
                break
            except ValueError:
                pass
        partQuantity = 0 - quantityAvailability(partId, partQuantity)
        fileUpdater("usedParts.txt", partId, "none", partQuantity, '-')
    #Process of suppling part
    else:
        while True:
            try:
                partQuantity = int(input("How many quantity supplied: "))
                break
            except ValueError:
                pass
        fileUpdater("suppliedParts.txt", partId, “none”, partQuantity, '+')
    #Update the available quanyity
    fileUpdater("parts.txt", partId, "none", partQuantity, '+')
    print("<Parts update successfully>\n")
    #Read a new part id then proceed to next iteration
```

```
        partId = input("Enter a part id to update(-1 to quit part update): ")
    return
#Assumed no negative integer input



def partTracking():
    #Check either any part recorded or not
    try:
        fileHandler = open("parts.txt", 'r')
        fileHandler.close()
    except FileNotFoundError:
        print("<No part recorded yet>")
        return
    displayTable("Part Tracking")
    operation = input("Choose an operation: ")
    #"-1" as sentinel value
    while operation != "-1":
        #Process of display all available quantity
        if operation == '1':
            fileHandler = open("parts.txt", 'r')
            data = fileHandler.readlines()
            fileHandler.close()
            for items in data:
                items = items.rstrip().split('\t')
                print("ID               :", items[0])
                print("Name             :", items[1])
                print("Quantity Available:", items[2])
                print("Warehouse Code   :", items[3])
                print("Assembly Section  :", items[4], '\n')
        #Process of display all short quantity that less than 10 units
```

```
    elif operation == '2':
        data = horizontalDataExtrator("parts.txt", 10, 2, '<', "true")
        if data == []:
            print("<No short quantity yet>")
            displayTable("Part Tracking")
            operation = input("Choose an operation: ")
            continue
        displayTable("Warehouse Code")
        warehouseCode = input("Enter a warehouse code(-1 to quit short quantity
tracking):
        ")
        while warehouseCode != "-1":
            warehouseCode = warehouseCode.upper()
            warehouseCode = codeExistanceChecker("Warehouse Code",
warehouseCode)
            newData = []
            for items in data:
                if items[3] == warehouseCode:
                    newData.append(items)
                    print("ID              :", items[0])
                    print("Name            :", items[1])
                    print("Quantity Available:", items[2])
                    print("Assembly Section  :", items[4], '\n')
            if newData == []:
                print("<No short quantity in this warehouse>")
            displayTable("Warehouse Code")
            warehouseCode = input("Enter a warehouse code(-1 to quit short quantity
             tracking): ")
    #Process of display every quantity used in assembly section
    elif operation == '3':
        try:
```

```
            fileHandler = open("usedParts.txt", 'r')
            data = fileHandler.readlines()
            fileHandler.close()
            displayTable("Warehouse Code")
            warehouseCode = input("Enter a warehouseCode(-1 to quit used quantity
tracking): ")
            while warehouseCode == "-1":
               warehouseCode = warehouseCode.upper()
               warehouseCode = codeExistanceChecker("Warehouse Code",
warehouseCode)
               newData = []
               for items in data:
                  items = items.rstrip().split('\t')
                  if items[3] == warehouseCode:
                     newData.append(items)
                     print("ID             :", items[0])
                     print("Name           :", items[1])
                     print("Quantity Used  :", items[2])
                     print("Assembly Section:", items[4], '\n')
               if newData == []:
                  print("<No used parts in this warehouse>")
               displayTable("Warehouse Code")
               warehouseCode = input("Enter a warehouseCode(-1 to quit used quantity
               tracking): ")
         except FileNotFoundError:
            print("<No used parts yet>")
      else:
         print("<Invalid Input>")
      displayTable("Part Tracking")
      operation = input("Choose an operation: ")
   return
```

```
def search():
    #Check either any part recorded or not
    try:
        fileHandler = open("parts.txt", 'r')
        fileHandler.close()
    except FileNotFoundError:
        print("<No part recorded yet>")
        return
    displayTable("Search")
    operation = input("Choose an operation: ")
    #"-1" as sentinel value
    while operation != "-1":
        #Process of display out all details by a part id
        if operation == '1':
            partId = input("Enter part ID you want to search(-1 to quit current search): ")
            while partId != "-1":
                partId = partId.upper()
                newData = []
                try:
                    dataQuantityAvailable = horizontalDataExtrator("parts.txt", partId, 0,
"==",
                    "false")
                    for count in range(3):
                        newData.append(dataQuantityAvailable[count])
                    dataQuantitySupplied = horizontalDataExtrator("suppliedParts.txt",
partId, 0,
                     "==", "false")
                    newData.append(dataQuantitySupplied[2])
                    try:
```

```
                dataQuantityUsed = horizontalDataExtrator("usedParts.txt", partId, 0,
"==",
                "false")
                newData.append(dataQuantityUsed[2])
            except FileNotFoundError:
                newData.append(0)
            for count in range(3,5):
                newData.append(dataQuantityAvailable[count])
            print("ID              :", newData[0])
            print("Name            :", newData[1])
            print("Quantity Available:", newData[2])
            print("Quantity Supplied :", newData[3])
            if newData[4] != 0:
                print("Quantity Used     :", newData[4])
            print("Warehouse Code    :", newData[5])
            print("Assembly Section  :", newData[6], '\n')
        except IndexError:
            print("<Part id does not exist>\n")
        partId = input("Enter part ID you want to search(-1 to quit current search):
")
    #Process of display out supplier details by a part id
    elif operation == '2':
        partId = input("Enter part ID you want to search(-1 to quit current search): ")
        while partId != "-1":
            partId = partId.upper()
            fileHandler = open("suppliedParts.txt", 'r')
            data = fileHandler.readlines()
            fileHandler.close()
            existance = "not exist"
            for items in data:
                items = items.rstrip().split('\t')
```

```
                if items[0] != partId:
                    continue
                items = horizontalDataExtrator("suppliers.txt", items[5], 0, "==", "false")
                print("ID              :", items[0])
                print("Name            :", items[1])
                print("Company Address :", items[2])
                print("Phone Number    :", items[3], '\n')
                existance = "exist"
                break
            if existance == "not exist":
                print("<Part id does not exist>\n")
            partId = input("Enter part ID you want to search(-1 to quit current search):
")
        #Process of display out all part details which supplied by a supplier id
        elif operation == '3':
            supplierId = input("Enter supplier ID you want to search(-1 to quit current
search): ")
            while supplierId != "-1":
                supplierId = supplierId.upper()
                data = horizontalDataExtrator("suppliedParts.txt", supplierId, 5, "==",
"true")
                if data == []:
                    print("<Supplier id does not exist OR this supplier haven't supply any
part
                    yet>\n")
                    supplierId = input("Enter supplier ID you want to search(-1 to quit
current
                    search): ")
                    continue
                for items in data:
                    print("ID              :", items[0])
                    print("Name            :", items[1])
```

```
                print("Quantity Supplied :", items[2])
                print("Warehouse Code    :", items[3])
                print("Assembly Section  :", items[4], '\n')
            supplierId = input("Enter supplier ID you want to search(-1 to quit current
search):
            ")
        else:
            print("<Invalid Input>")
        displayTable("Search")
        operation = input("Choose an operation: ")
    return



def menu():
    #menu that let user to choose operation
    displayTable("Menu")
    operation = input("Choose an operation: ")
    while operation != "-1":
        if operation == '1':
            partsCreation("none")
        elif operation == '2':
            suppliersCreation("none")
        elif operation == '3':
            partUpdate()
        elif operation == '4':
            partTracking()
        elif operation == '5':
            search()
        else:
            print("<Invalid Input>")
        displayTable("Menu")
```

```
        operation = input("Choose an operation: ")
    return

#Generate table in a file before enter menu
generateTable()
menu()
```

3.1 Explanation

3.1.1 Start of the program

First and foremost, the program will start with calling generateTable function to Generate table in a file before enter menu. In the generateTable the function will declare all required table as data then create table.txt file and write the data into it, it any new car model and assembly section will be editing from here which is assumed rarely going to happen. After that, the user will enter to menu function which will first call displayTable function to display valid operation in menu function to let user to choose. In displayTable function, it will read table.txt file that generated just now as data then find out the line which is equal to table name and END table in order to display the table out to the user.

3.1.2 Data extractor

There are two data extractors in this program which is verticalDataExtractor and horizontalDataExtractor. Firstly, verticalDataExtractor is to extract vertical data in desired file into a list and return to the function which called it, while it also can convert the element in the list to integer if wanted. Besides that, horizontalDataExtrator is to extract horizontal data in desired file into a list or a variable and return to the function which called it. There are 2 condition for horizontalDataExtrator which is equality and less than, the condition will used to compare between element of line in data in desired file and targeted data.

3.1.3 Record and Rearrange

After all data is stored in a master list, then the function will call for record function which is to write the element in line of data one by one into desired file separated with TAB and NEWLINE for every line. If the record function is called in ‘a’ (append mode) then it will also call for rearrange function. The rearrange will first get all ids which originally existed in the desired file in integer form using verticalDataExtractor, then sorted it with ascending order. After that, the arranged ids will be compared with all 1st element in line of data from desired file, if they are same then the line will be stored inside a new array. Lastly, it will call for record function in ‘w’ (write mode) to rewrite the file by id with ascending order.

3.1.4 Existence Checker

There are two existence checkers in this program which is idExistanceChecker and codeExistanceChecker. Firstly, idExistanceChecker is to check whether the id is existed in desired file or not, while it also can directly link to respective creation function if the function which called it want to. Besides that, codeExistanceChecker is to check the validation of a code that existed in a desired table, this function will find out the list of valid code above table name in table.txt file, then check is it valid, if not then the function will then request user to input a code again until the code is valid then return to the function which called it

3.1.5 Quantity Availability

There is a function called quantityAvailability which is used to check the quantity availability when a user is request part from warehouse to use in assembly section, while it will also make sure the quantity input by user is valid and in integer form or else it will keep asking for new quantity, if the quantity is exceed available quantity it will also request user input a new quantity again and call for itself until it is available then return the confirmed quantity to the function which called it.

3.1.6 File Updater

There is a function called fileUpdater which can either add a new data or update the quantity of an existed data in a file. It is able to update desired file and with desired operation. If the data with same part id already existed than it will loop all the line in data and calculate the new quantity when encounter the same part id, then append to the new array to call for record function to rewrite it. Else if the data with same part id is not existed, then it will calculate the data with 0-partQuantity then store inside an array and call for record function in append mode and also rearrange desired file to make sure the data is sorted in ascending order.

3.1.7 Creations

There are two type creation which are part creation and supplier creation, they both will possibly receive a id from idExistanceChecker and run for one time without any part id input. Besides that, both of them operate quite similar which is store all data inside a small list then small list into a master list then call for record and rearrange function. All id will be call for idExistanceChecker and check for the format to confirm the validation. Master will be store instantly without waiting until all data is input because it is to prevent user to input same id before record it into the file, while the master list is to follow the algorithm designed in record function.

3.1.8 Part Update

This is a function that allow user to update the part quantity either increase quantity or decrease quantity. Firstly, it will check for validation and existance of part id that input. If the id is not existance then it will directly link to part creation to fill all the details. For decrease quantity, it will call for quantityAvailability function to check the validation of the quantity until it is valid. For both increase and decrease it will chech for the quantity either it is in integer form or not. Lastly it will call for fileUpdate function to update related file.

3.1.8 Part Tracking and Searching Functionalities

Both of the function is to display data from the text file, it will check for the existence of a particular file before executing the instructions. Firstly, all operation inside both function will keep looping until user input a sentinel value. Besides that, it will not have any case sensitive when input id or code. Moreover, any wrong id input will let the program to display "<id does not exist>" without checking the format of it.

4. Screenshots of sample input/output and explanation

4.1 Menu

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

This is the first interface that display to user when start the program. There are 6 valid operation number to input which will call respective function excluding "-1" is to quit the program entirely. When an invalid input is read, then the program will react as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 6
<Invalid Input>

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

Which mean it will tell the user that operation number doesn't exist, then it will display the menu table again and request the user to choose an operation.

4.1 Parts Creation

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 1
Enter a unique part id(-1 to quit parts creation): P0001
Enter a new part name: Black Oil
Enter the initial quantity for that part: 20

---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Please enter the warehouse code: WBS

---------------------------------
|  Section              |  Code  |
---------------------------------
|  Engine Section       |  ES    |
|  Body Work Section    |  BWS   |
|  Air-con Section      |  AS    |
---------------------------------
Please enter the assembly section code they are used: ES
Enter a supplier id who supply this part: S0001
<New Id Detected>

Enter new supplier name: SIA
Enter supplier company address: Jalan 7/40, Taman Bunga
Enter supplier phone number: 016-6956139
<Supplier created successfully>

<Part created successfully>

Enter a unique part id(-1 to quit parts creation):
```

Figure above shows how a full run through in parts creation operate without create a supplier first. When first enter the part creation, the program will request user to input a unique part id and there is some validation for the id which are part id must start with character ‘p’ with no case sensitive together with 4 digits of numbers. This validation will be automatically checked by the program as shown:

```
Choose an operation: 1
Enter a unique part id(-1 to quit parts creation): p1
<Invalid Id>
<Part Id must start with character 'p' together with 4 number>

Enter a unique part id(-1 to quit suppliers creation): s0001
<Invalid Id>
<Part Id must start with character 'p' together with 4 number>

Enter a unique part id(-1 to quit suppliers creation):
```

It will display "<ID Existed>" and notice the user that part id must be unique. Moreover, the validation of quantity will also be checking the validation either it is a integer or not as shown:

```
Enter a unique part id(-1 to quit parts creation): P0001
<ID Existed>

Please enter a UNIQUE part id(-1 to quit parts creation):
```

If a invalid quantity is input the program will react as shown:

```
Enter the initial quantity for that part: aaa
<Invalid Quantity>

Enter the initial quantity for that part: 3.1412
<Invalid Quantity>
```

The program will keep request user to input quantity until it is valid. In addition, warehouse code and assembly section will also check either it is existed in the table or not with no case sensitive as shown:

```
---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Please enter the warehouse code: aaa
<Invalid Input>
Please enter a valid code: wbs

---------------------------------
|  Section              |  Code  |
---------------------------------
|  Engine Section       |  ES    |
|  Body Work Section    |  BWS   |
|  Air-con Section      |  AS    |
---------------------------------
Please enter the assembly section code they are used: aaa
<Invalid Input>
Please enter a valid code: es
Enter a supplier id who supply this part:
```

Next, since the supplier id is required to input when part creation, then if the supplier id is already register, the program will react as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 1
Enter a unique part id(-1 to quit parts creation): P0001
Enter a new part name: Black Oil
Enter the initial quantity for that part: 20

---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Please enter the warehouse code: WBS

---------------------------------
|  Section              |  Code  |
---------------------------------
|  Engine Section       |  ES    |
|  Body Work Section    |  BWS   |
|  Air-con Section      |  AS    |
---------------------------------
Please enter the assembly section code they are used: ES
Enter a supplier id who supply this part: S0001
<Part created successfully>

Enter a unique part id(-1 to quit parts creation):
```

Last but not least, if the user input “-1” when the program is asking for part id, then the program will return to menu as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 1
Enter a unique part id(-1 to quit parts creation): -1

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

4.2 Suppliers Creation

```
--------------------------------
|  No.  |  Operation           |
--------------------------------
|  1.   |  Parts Creation      |
|  2.   |  Suppliers Creation  |
|  3.   |  Part Update         |
|  4.   |  Part Tracking       |
|  5.   |  Search Function     |
| -1.   |  Quit Program        |
--------------------------------
Choose an operation: 2
Enter a unique supplier id(-1 to quit suppliers creation): S0001
Enter new supplier name: SIA
Enter supplier company address: Jalan 7/40, Taman Bunga
Enter supplier phone number: 061-6956139
<Supplier created successfully>

Enter a unique supplier id(-1 to quit suppliers creation):
```

Figure above shows how a full run through in suppliers creation operate. When first enter the part creation, the program will request user to input a unique supplier id and there is some validation for the id which are part id must start with character ‘s’ with no case sensitive together with 4 digits of numbers. This validation will be automatically checked by the program as shown:

```
Enter a unique supplier id(-1 to quit suppliers creation): S1
<Invalid Id>
<Supplier Id must start with character 's' together with 4 number>

Enter a unique supplier id(-1 to quit suppliers creation): P0001
<Invalid Id>
<Supplier Id must start with character 's' together with 4 number>

Enter a unique supplier id(-1 to quit suppliers creation):
```

It will display “<ID Existed>” and notice the user that part id must be unique as shown:

```
Enter a unique supplier id(-1 to quit suppliers creation): S0001
<ID Existed>

Please enter a UNIQUE supplier id(-1 to quit suppliers creation):
```

Lastly, if the user input “-1” when the program is asking for supplier id, then the program will return to menu as shown:

```
--------------------------------
|  No.  |  Operation           |
--------------------------------
|  1.   |  Parts Creation      |
|  2.   |  Suppliers Creation  |
|  3.   |  Part Update         |
|  4.   |  Part Tracking       |
|  5.   |  Search Function     |
| -1.   |  Quit Program        |
--------------------------------
Choose an operation: 2
Enter a unique supplier id(-1 to quit suppliers creation): -1

--------------------------------
|  No.  |  Operation           |
--------------------------------
|  1.   |  Parts Creation      |
|  2.   |  Suppliers Creation  |
|  3.   |  Part Update         |
|  4.   |  Part Tracking       |
|  5.   |  Search Function     |
| -1.   |  Quit Program        |
--------------------------------
Choose an operation:
```

4.3 Part Update

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 3
Enter a part id to update(-1 to quit part update): P0001
Enter any key to supply the part(-1 to use in assembly section):
How many quantity supplied: 10
<Parts update successfully>

Enter a part id to update(-1 to quit part update):
```

Figure above shows how a full run through in increasing quantity of part update operate. When first enter the part creation, the program will request user to input a part id either the part id is existed or not and there is some validation for the id which are part id must start with character ‘p’ with no case sensitive together with 4 digits of numbers. This validation will be automatically checked by the program as shown:

```
Enter a part id to update(-1 to quit part update): p1
<Invalid Id>
<Part Id must start with character 'p' together with 4 number>

Enter a part id to update(-1 to quit part update): s0001
<Invalid Id>
<Part Id must start with character 'p' together with 4 number>

Enter a part id to update(-1 to quit part update):
```

If the part id has not register yet, then the program will automatically enter to part creation first to fill in all the details as shown:

```
Enter a part id to update(-1 to quit part update): P0001
Enter a new part name: Black Oil
Enter the initial quantity for that part: 20

---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Please enter the warehouse code: WBS

---------------------------------
|  Section              |  Code  |
---------------------------------
|  Engine Section       |  ES    |
|  Body Work Section    |  BWS   |
|  Air-con Section      |  AS    |
---------------------------------
Please enter the assembly section code they are used: ES
Enter a supplier id who supply this part: S0001
<New Id Detected>

Enter new supplier name: SIA
Enter supplier company address: Jalan 7/40, Taman Bunga
Enter supplier phone number: 016-6956139
<Supplier created successfully>

<Part created successfully>

Enter any key to supply the part(-1 to use in assembly section):
```

Besides that, if the user wants to request for part to use in assembly section then there will be a quantity availability and validation check as shown:

```
Enter a part id to update(-1 to quit part update): P0001
Enter any key to supply the part(-1 to use in assembly section): -1
Available quantity: 20
How many quantity wanted to use: 21
<Quantity exceed available quantity>
How much quantity wanted to use: 65.54
How much quantity wanted to use:
```

The program will keep ask for quantity until the quantity is valid and available, then it will request for next part id to update as shown:

```
How much quantity wanted to use: 1
<Parts update successfully>

Enter a part id to update(-1 to quit part update):
```

Lastly, if the user input “-1” when the program is asking for part id, then the program will return to menu as shown:

```
Enter a part id to update(-1 to quit part update): -1

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

4.4 Part Tracking

When first enter the part tracking, the program will request user to choose an operation number while "-1" to quit part tracking and it will automatically detect either the input operation number is existed on the table or not as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Available Quantity   |
|  2.   |  Short Quantity       |
|  3.   |  Used Quantity        |
| -1.   |  Quit Part Tracking   |
---------------------------------
Choose an operation: 4
<Invalid Input>

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Available Quantity   |
|  2.   |  Short Quantity       |
|  3.   |  Used Quantity        |
| -1.   |  Quit Part Tracking   |
---------------------------------
Choose an operation: -1

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

If still does not have any part recorded yet then the program will return to menu as shown:

```
--------------------------------
|  No.  |  Operation           |
--------------------------------
|  1.   |  Parts Creation      |
|  2.   |  Suppliers Creation  |
|  3.   |  Part Update         |
|  4.   |  Part Tracking       |
|  5.   |  Search Function     |
| -1.   |  Quit Program        |
--------------------------------
Choose an operation: 4
<No part recorded yet>

--------------------------------
|  No.  |  Operation           |
--------------------------------
|  1.   |  Parts Creation      |
|  2.   |  Suppliers Creation  |
|  3.   |  Part Update         |
|  4.   |  Part Tracking       |
|  5.   |  Search Function     |
| -1.   |  Quit Program        |
--------------------------------
Choose an operation:
```

If the user choose operation ‘1’ then all part available quantity will be display, then return to part tracking menu again as shown:

```
---------------------------------
|  No. |  Operation             |
---------------------------------
|  1.  |  Available Quantity    |
|  2.  |  Short Quantity        |
|  3.  |  Used Quantity         |
| -1.  |  Quit Part Tracking    |
---------------------------------
Choose an operation: 1
ID                : P0001
Name              : Black Oil
Quantity Available: 30
Warehouse Code    : WBS
Assembly Section  : ES

ID                : P0002
Name              : Aluminium Plate
Quantity Available: 40
Warehouse Code    : WAY
Assembly Section  : BWS

ID                : P0003
Name              : Cooling System Module
Quantity Available: 60
Warehouse Code    : WBY
Assembly Section  : AS


---------------------------------
|  No. |  Operation             |
---------------------------------
|  1.  |  Available Quantity    |
|  2.  |  Short Quantity        |
|  3.  |  Used Quantity         |
| -1.  |  Quit Part Tracking    |
---------------------------------
Choose an operation:
```

If the user choose operation ‘2’ then it will first request use to input a warehouse code while the program will automatically check the validation of the code with no case sensitive, next, all part with short quantity which is less than 10 units will be display by warehouse, then request user to input a warehouse code again as shown:

```
---------------------------------
|  No. |  Operation             |
---------------------------------
|  1.  |  Available Quantity    |
|  2.  |  Short Quantity        |
|  3.  |  Used Quantity         |
| -1.  |  Quit Part Tracking    |
---------------------------------
Choose an operation: 2

---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Enter a warehouse code(-1 to quit short quantity tracking): aaa
<Invalid Input>
Please enter a valid code: wbs
ID                : P0001
Name              : Black Oil
Quantity Available: 4
Assembly Section  : ES


---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Enter a warehouse code(-1 to quit short quantity tracking):
```

If the user input “-1” when the program asking for a warehouse code, then it will return to part tracking menu again as shown:

```
---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Enter a warehouse code(-1 to quit short quantity tracking): -1

---------------------------------
|  No. |  Operation             |
---------------------------------
|  1.  |  Available Quantity    |
|  2.  |  Short Quantity        |
|  3.  |  Used Quantity         |
| -1.  |  Quit Part Tracking    |
---------------------------------
Choose an operation:
```

If the user choose operation '3' then it will first request use to input a warehouse code while the program will automatically check the validation of the code with no case sensitive, next, all used part quantity and its details will be display by warehouse, then request user to input a warehouse code again as shown:

```
---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Enter a warehouseCode(-1 to quit used quantity tracking): aaa
<Invalid Input>
Please enter a valid code: wbs
ID              : P0002
Name            : Black Oil
Quantity Used   : 10
Assembly Section: ES


---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Enter a warehouseCode(-1 to quit used quantity tracking):
```

If the user input "-1" when the program asking for a warehouse code, then it will return to part tracking menu again as shown:

```
---------------------------------
|  Model                |  Code  |
---------------------------------
|  Bios                 |  WBS   |
|  Ambry                |  WAY   |
|  Barrier              |  WBR   |
---------------------------------
Enter a warehouseCode(-1 to quit used quantity tracking): -1

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Available Quantity   |
|  2.   |  Short Quantity       |
|  3.   |  Used Quantity        |
| -1.   |  Quit Part Tracking   |
---------------------------------
Choose an operation:
```

4.5 Searching Functionalities

When first enter the part tracking, the program will request user to choose an operation number while "-1" to quit search and it will automatically detect either the input operation number is existed on the table or not as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 5

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Part Record          |
|  2.   |  Supplier Detail      |
|  3.   |  Supplier Supplied    |
| -1.   |  Quit Search          |
---------------------------------
Choose an operation: 4
<Invalid Input>

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Part Record          |
|  2.   |  Supplier Detail      |
|  3.   |  Supplier Supplied    |
| -1.   |  Quit Search          |
---------------------------------
Choose an operation: -1

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

If still does not have any part recorded yet then the program will return to menu as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 5
<No part recorded yet>

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation:
```

If the user choose operation ‘1’ then the program will request use to input a part id while the program will check the existence of the part id and display all its details and request part id again to next search as shown:

```
---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Parts Creation       |
|  2.   |  Suppliers Creation   |
|  3.   |  Part Update          |
|  4.   |  Part Tracking        |
|  5.   |  Search Function      |
| -1.   |  Quit Program         |
---------------------------------
Choose an operation: 5

---------------------------------
|  No.  |  Operation            |
---------------------------------
|  1.   |  Part Record          |
|  2.   |  Supplier Detail      |
|  3.   |  Supplier Supplied    |
| -1.   |  Quit Search          |
---------------------------------
Choose an operation: 1
Enter part ID you want to search(-1 to quit current search): P0001
ID                : P0001
Name              : Black Oil
Quantity Available: 30
Quantity Supplied : 30
Warehouse Code    : WBS
Assembly Section  : ES

Enter part ID you want to search(-1 to quit current search):
```

If the user input a id that does not exist then the program will ask for another id again to enter the next loop as shown:

```
Enter part ID you want to search(-1 to quit current search): P9999
<Part id does not exist>

Enter part ID you want to search(-1 to quit current search):
```

If the user choose operation '2' then the program will request use to input a part id while the program will check the existence of the part id and display all its details and request part id again to next search as shown:

```
--------------------------------
|  No. |  Operation             |
--------------------------------
|  1.  |  Part Record           |
|  2.  |  Supplier Detail       |
|  3.  |  Supplier Supplied     |
| -1.  |  Quit Search           |
--------------------------------
Choose an operation: 2
Enter part ID you want to search(-1 to quit current search): P0001
ID              : S0001
Name            : SIA
Company Address : Jalan 7/40, Taman Bunga
Phone Number    : 016-6956139

Enter part ID you want to search(-1 to quit current search):
```

If the user input a id that does not exist then the program will ask for another id again to enter the next loop as shown:

```
Enter part ID you want to search(-1 to quit current search): P9999
<Part id does not exist>

Enter part ID you want to search(-1 to quit current search):
```

If the user choose operation '3' then the program will request use to input a part id while the program will check the existence of the part id and display all its details and request part id again to next search as shown:

```
---------------------------------
|  No. |  Operation             |
---------------------------------
|  1.  |  Part Record           |
|  2.  |  Supplier Detail       |
|  3.  |  Supplier Supplied     |
| -1.  |  Quit Search           |
---------------------------------
Choose an operation: 3
Enter supplier ID you want to search(-1 to quit current search): S0001
ID                : P0001
Name              : Black Oil
Quantity Supplied : 10
Warehouse Code    : WBS
Assembly Section  : ES

ID                : P0004
Name              : Aluminium Plate
Quantity Supplied : 40
Warehouse Code    : WBS
Assembly Section  : BWS

ID                : P0007
Name              : Medium-quality Black Oil
Quantity Supplied : 10
Warehouse Code    : WAY
Assembly Section  : ES

ID                : P0010
Name              : Car door
Quantity Supplied : 40
Warehouse Code    : WAY
Assembly Section  : BWS

ID                : P0013
Name              : High-quality Black Oil
Quantity Supplied : 10
Warehouse Code    : WBY
Assembly Section  : ES

ID                : P0016
Name              : Bumper
Quantity Supplied : 40
Warehouse Code    : WBY
Assembly Section  : BWS

Enter supplier ID you want to search(-1 to quit current search):
```

If the user enter a supplier id with no data then the program will react as shown:

```
Enter supplier ID you want to search(-1 to quit current search): S9999
<Supplier id does not exist OR this supplier haven't supply any part yet>

Enter supplier ID you want to search(-1 to quit current search):
```

Conclusion

In a nutshell, a program that meet all the requirements is created while supplier creation function is added to the program. The automobile manufacturing plant can now use the program to manage automobile parts in all the warehouses if they meet all the assumptions. In my opinion, there are still some improvements to the program to remove few of the assumptions in order to let the program more user friendly, this program is not mature enough for user who does not have a fundamental knowledge of programming to use.