INDIVIDUAL ASSIGNMENT

Introduction To C Programming

APD1F2106CS(CYB)

HAND IN DATE : 28th February 2022

WEIGHTAGE : 50%

LECTURER : NUR FATINI BINTI ISMAIL

STUDENT NAME : CHANG SHIAU HUEI

TP NUMBER : TP060322

TABLE OF CONTENTS

INTRODUCTION

In Sungai Tunas, an automobile manufacturing plant has been assembling different models of passenger cars. After some consideration due to staff shortages, they decided to utilise an Automobile Parts Inventory Management System for all warehouses in its assembly divisions. Hence, this program written in C acts as a menu-driven Automobile Parts Inventory Management System to automate certain processes in handling the logistics of automobile parts inventory. It consists of 4 main functions, namely for parts inventory creation, updating the created part inventories, tracking the created part inventories and searching each part's records as well as their respective supplier details. By entering information into Section 1, the input will be written onto 2 files, one being "supplier.txt" and another being one out of three files which are "WBZ.txt", "WSL.txt" and "WAR.txt" depending on the part's model that was provided by the user. "WBZ.txt", "WSL.txt" and "WAR.txt" will consist of relevant details about the parts such as a unique part ID, part name, part quantity, respective supplier, and status of the part, with "LOW" indicating that the quantity of the part is less than 10, while "NORMAL" indicating that the quantity of the part is 10 or more. "supplier.txt" will then consist of information related to the supplier of the part. For instance, a unique supplier ID, supplier name, and the part ID of the provided part.

ASSUMPTIONS

1. Users cannot access Section 2 to 4 if “supplier.txt” is empty (Which means “WBZ.txt”, “WSL.txt” and “WAR.txt” are empty as well), in this case, only Section 1 can be chosen.
2. Part name, supplier name, Part ID, and Supplier ID length should be only within 99 characters.
3. Most incorrect inputs in the program (e.g. datatype, length, options that are not shown in menus, repetition of input, entry exists or does not exists) will be validated.
4. Part name and supplier name length can consist of numbers, as the user might insert the serial codes of parts instead of the name, and a supplier might have a number in their name (e.g. Company123).
5. Two suppliers cannot have the same name in “supplier.txt”.
6. A supplier can supply more than one part, but a part cannot be supplied by more than one supplier.
7. There can only be distinct types of parts by only one supplier each in each warehouse file (“WBZ.txt”, “WSL.txt” and “WAR.txt”).
8. If the quantity of a part is 0, it will remain in the warehouse files and “supplier.txt”. The admin would have to manually go to the files to delete the record completely if he wants to.
9. Users can quit the program almost at any part of the process.
10. Capitalization in inputs does not influence the output.
11. The arrangement of data in “WBZ.txt”, “WSL.txt” and “WAR.txt” will always be in ascending order according to the number that follows after the 3 letter code.
12. The arrangement of data in “supplier.txt” follows an order by the latest supply update.

EXTRA FEATURES

Each supplier has a unique Supplier ID allocated to them which consists of the word “SUP” and a number. The supplier ID will be placed into the warehouse files instead of the supplier names.

DESIGN OF THE PROGRAM

PSEUDOCODE

```
PROGRAM AutomobilePartsManagementSystem
BEGIN
    INCLUDE <stdio.h>
    INCLUDE <stdlib.h>
    INCLUDE <string.h>
    INCLUDE <unistd.h>
    INCLUDE <stdbool.h>

    DEFINE FILENAME_SIZE to 100
    DEFINE MODEL_SIZE to 100
    DEFINE WAREHOUSE_CHOICE_SIZE to 100
    DEFINE PART_ID_SIZE to 100
    DEFINE PART_NAME_SIZE to 100
    DEFINE SUP_ID_SIZE to 100
    DEFINE SUP_NAME_SIZE to 100
    DEFINE MAX_SIZE_OF_SUPPLIER_NO to 6
    DEFINE MAX_SIZE_OF_PART_NO to 6
    DEFINE MAX_PER_LINE_SIZE to 200
    DEFINE MAX_SIZE_OF_QUANTITY to 6

    DEFINE Warehouse STRUCTURE as warehouse
        DECLARE VARIABLE warehouseChoice[WAREHOUSE_CHOICE_SIZE],
        partID[PART_ID_SIZE], partName[PART_NAME_SIZE] as char
        DECLARE VARIABLE partQuantity as int
    ENDSTRUCTURE

    DEFINE Supplier STRUCTURE as supplier
        DECLARE VARIABLE supID[SUP_ID_SIZE], supName[SUP_NAME_SIZE as char
    ENDSTRUCTURE
```

```
FUNCTION printingSeparator(DECLARE length as int, DECLARE item as char)
        LOOP (DECLARE i as int) FROM 0 TO < length STEP 1
                Display item
                NEXT i
        END LOOP
        RETURN
ENDFUNCTION

FUNCTION lineCount(DECLARE fileUsed[FILENAME_SIZE] as char)
        DECLARE lineCount as int
        lineCount = 0
        DECLARE chr as char
        DECLARE fptr as file pointer
        OPEN fileUsed with READ mode AS fptr
                char = getc(fptr)
                WHILE char != EOF
                        IF (chr == newline) THEN
                                lineCount = lineCount + 1
                        ENDIF
                        char = getc(fptr)
                ENDWHILE
        CLOSE fptr
        RETURN lineCount
ENDFUNCTION

FUNCTION seekToNextLine()
        DECLARE VARIABLE c as int
        WHILE (c = fgetc(stdin)) != EOF AND c != newline)
        ENDWHILE
        RETURN
ENDFUNCTION

FUNCTION messageCentered(DECLARE VARIABLE message as char pointer)
```

```
        DECLARE VARIABLE length, position as int
        length = (69 - strlen(message)) / 2
        LOOP position FROM 0 TO < length STEP 1
                Display “ ”
                NEXT position
        ENDLOOP
        Display message
        RETURN
ENDFUNCTION

FUNCTION exitingProgram(DECLARE VARIABLE status as int)
        IF (status == 1) THEN
                Display "Thank you for using this system :D"
        ENDIF
        IF (status == 2) THEN
                Display "Entry does not exist. Exiting program..."
        ENDIF
        IF (status == 3) THEN
                Display "Invalid input. Exiting program..."
        ENDIF
        IF (status == 4) THEN
                Display "Error opening files(s). Exiting program..."
        ENDIF
        IF (status == 5) THEN
                Display "Entry already exists. Exiting program..."
        ENDIF
        IF (status == 6) THEN
                Display "No records exist. Exiting system..."
        ENDIF
        printingSeparator(69, '=')
        Exit program
        RETURN
ENDFUNCTION
```

```
FUNCTION entryExists(DECLARE VARIABLE entry as char pointer, file as file pointer)
        DECLARE VARIABLE partFinder as int
        partFinder = 0
        DECLARE VARIABLE lineInFile[MAX_PER_LINE_SIZE] as char
        WHILE Read lineInFile from file != NULL
                IF entry matches lineInFile THEN
                        partFinder = partFinder + 1
                ENDIF
                IF (partFinder == 1) THEN
                        RETURN partFinder
                ENDIF
        ENDWHILE
        RETURN partFinder
ENDFUNCTION

FUNCTION entryCheckForExistencePartID (DECLARE VARIABLE entryPartID as char
pointer)
        DECLARE VARIABLE flag as int
        flag = 0
        WHILE true
                Display "Please insert your Part ID. Enter 'X' to quit:”
                Read entryPartID
                Change entryPartID to uppercase
                seekToNextLine()
                IF entryPartID is X THEN
                        exitingProgram(1)
                ELSE
                        IF length of entryPartID > PART_ID_SIZE THEN
                                Display “Please input your Part ID within”, PART_ID_SIZE,
                                “characters."
                                CONTINUE
                        ELSE
```

```
                    IF (lineCount("supplier.txt") > 0) THEN
                        DECLARE VARIABLE warehouse[PART_ID_SIZE] as
                        char
                        Extract the first 3 characters in entryPartID to warehouse
                        DECLARE VARIABLE fileName[FILENAME_SIZE] as
                        char
                        Concatenate warehouse and ".txt" to fileName
                        DECLARE VARIABLE fptr as file pointer
                        OPEN fileName with READ mode AS fptr
                            flag = entryExists(entryPartID, fptr)
                        CLOSE fptr
                        IF (flag == 1) THEN
                            IF fileName is "WBZ.txt" THEN
                                RETURN "WBZ.txt"
                            ENDIF
                            IF fileName is "WSL.txt" THEN
                                RETURN "WSL.txt"
                            ENDIF
                            IF fileName is "WAR.txt" THEN
                                RETURN "WAR.txt"
                            ENDIF
                        ELSE
                            exitingProgram(2)
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDWHILE
ENDFUNCTION

FUNCTION entryCheckForDuplicate(DECLARE VARIABLE fileUsed[FILENAME_SIZE],
item[10] as char , maxSizeOfItem as int, entry as char pointer)
    DECLARE VARIABLE flag as int
```

```
        flag = 1
        DECLARE VARIABLE fptr as file pointer
        OPEN fileUsed with READ mode AS fptr
        WHILE true
                Display “Please insert the”, item, “that you want to register. Enter 'X' to quit:”
                Read entry
                Change entry to uppercase
                seekToNextLine()
                IF entry is X THEN
                        exitingProgram(1)
                ELSE
                        IF length of entry > maxSizeOfItem THEN
                        Display "Please input a”, item, “within”, maxSizeOfItem, “characters."
                        CONTINUE
                        ELSE
                                IF (lineCount(fileUsed) > 0) THEN
                                        flag = entryExists(entry, fptr)
                                        CLOSE fptr
                                        IF (flag == 0) THEN
                                                BREAK
                                        ELSE
                                                exitingProgram(5)
                                        ENDIF
                                ENDIF
                        ENDIF
                ENDIF
        ENDWHILE
        RETURN
ENDFUNCTION

FUNCTION replaceLine(DECLARE VARIABLE fileUsed[FILENAME_SIZE],
replaceText[MAX_PER_LINE_SIZE] as char, lineNum as int)
        DECLARE VARIABLE filePtr, tempPtr as file pointer
```

```
        DECLARE VARIABLE buffer[MAX_PER_LINE_SIZE] as char
        OPEN fileUsed with READ mode AS filePtr
        OPEN "tempFile.txt" with WRITE mode AS tempPtr
            IF (filePtr == NULL) OR (tempPtr == NULL)
                exitingProgram(4)
            ENDIF
            DECLARE VARIABLE keepReading as bool
            keepReading = true
            DECLARE VARIABLE currentLine as int
            currentLine = 1
            DOWHILE keepReading
                Read buffer from filePtr
                IF filePtr reaches EOF THEN
                    keepReading = false
                ELSE
                    IF (currentLine == lineNum) THEN
                        Write tempPtr into replaceText
                    ELSE
                        Write tempPtr into buffer
                    ENDIF
                ENDIF
                currentLine = currentLine + 1
            ENDDO
        CLOSE filePtr, tempPtr
        Remove fileUsed
        Rename fileUsed to "tempFile.txt"
        RETURN
ENDFUNCTION

FUNCTION updateStockQuantity(DECLARE VARIABLE partID[PART_ID_SIZE],
partName[PART_NAME_SIZE], quantity[MAX_SIZE_OF_QUANTITY],
supID[SUP_ID_SIZE], fileUsed[FILENAME_SIZE] as char, lineNum as int)
        DECLARE VARIABLE currentQuantity, result, inputNum as int
```

```
Convert quantity to characters
currentQuantity = quantity
WHILE true
        DECLARE VARIABLE choice as int
        Display "Pick an operation by entering the corresponding number:"
        Display "1 - Increase stock quantity”
        Display “2 - Decrease stock quantity"
        Display "Enter negative numbers to quit:"
        Read choice
        seekToNextLine()
        IF (choice > 0) AND (choice < 3) THEN
                IF (choice == 1) THEN
                        Display "Please insert the stock amount to be added:"
                        Read inputNum
                        seekToNextLine()
                        result = currentQuantity + inputNum
                ENDIF
                IF (choice == 2) THEN
                        Display "Please insert the stock amount to be subtracted: "
                        Read inputNum
                        seekToNextLine()
                        IF (inputNum > currentQuantity) THEN
                                Display "Current stock is insufficient, unable to grant parts
                                for assembly. Please try again."
                                CONTINUE
                        ENDIF
                        result = currentQuantity - inputNum
                ENDIF
                Display "Details updated. Returning to main menu..."
                BREAK
        ELSE
                IF (choice == 0) OR (choice > 2) THEN
                        Display "Please choose a number from 1 to 2 only."
```

```
                ELSE
                        exitingProgram(1)
                ENDIF
        ENDIF
    ENDWHILE
    DECLARE VARIABLE replaceText[MAX_PER_LINE_SIZE] as char
    IF (result < 10) THEN
        Write formatted output to replaceText “%s|%s|%d|%s|LOW\n",partID, partName,
        result, supID
    ELSE
        Write formatted output to replaceText “%s|%s|%d|%s|NORMAL\n",partID,
        partName, result, supID
    ENDIF
     replaceLine(fileUsed, lineNum, replaceText)
    RETURN
ENDFUNCTION

FUNCTION lineNumberFinderAndPrintResults(DECLARE VARIABLE
fileUsed[FILENAME_SIZE], ID[PART_ID_SIZE], extraPartForSection1[PART_ID_SIZE] as
char, choice as int)
    DECLARE VARIABLE filePtr as file pointer
    DECLARE VARIABLE lineNum, findPart as int
    lineNum = 0
    findPart = 0
    DECLARE VARIABLE lineInFile[MAX_PER_LINE_SIZE] as char
    OPEN fileUsed with READ mode AS filePtr
        WHILE (Read buffer from lineInFile != NULL)
            IF ID is in lineInFile THEN
                findPart = findPart + 1
            ENDIF
            lineNum = lineNum + 1
            IF (findPart == 1) THEN
                BREAK
```

```
                ENDIF
            ENDWHILE
CLOSE filePtr
DECLARE VARIABLE x as int
x = 0
DECLARE VARIABLE array[5] as char pointer
lineInFile[strcspn(lineInFile, "newline")] = 0
DECLARE VARIABLE token as char pointer
Breaks lineInFile into a series of tokens using a “|” delimiter
token[strcspn(token, "newline")] = 0
WHILE (token != NULL)
    array[x] = token
    x = x + 1
    token = strtok(NULL, "|")
ENDWHILE
Display newline
printingSeparator(69, '-')
Display newline
IF (choice == 1) OR (choice == 3) THEN
        messageCentered("Part Details")
        Display newline
        printingSeparator(69, '-')
        Display “Part ID =”, array[0]
        Display “Part Name =”, array[1]
        Display “Part Quantity =”, array[2]
        Display “Part Status =”, array[4]
        Display “Supplier ID =”, array[3]
        printingSeparator(69, '-')
        IF (choice == 3) THEN
                updateStockQuantity(array[0], array[1], array[2], array[3], fileUsed,
                lineNum)
        ENDIF
ENDIF
```

```
        IF (choice == 2) THEN
                messageCentered("Supplier Details")
                Display newline
                printingSeparator(69, '-')
                Display “Part ID =”, array[2]
                Display “Supplier ID =”, array[0]
                Display “Supplier Name =”, array[1]
                printingSeparator(69, '-')
        ENDIF
        IF (choice == 4) THEN
                OPEN fileUsed with APPEND mode AS filePtr
                        Write formatted output to filePtr "%s|%s|%s\n", array[0], array[1],
                        extraPartForSection1
                CLOSE filePtr
        ENDIF
        RETURN
ENDFUNCTION

FUNCTION selectingWarehouse()
        DECLARE VARIABLE choice as int
        Display “Pick a model by entering the corresponding number:"
        Display "1 - Blaze (BZ)”
        Display “2 - Silk (SL)”
        Display “3 - Armer (AR)"
        WHILE true
                Display "Enter negative numbers to quit:"
                Read choice
                seekToNextLine()
                IF (choice > 0) AND (choice < 4) THEN
                        IF (choice == 1) THEN
                                RETURN "WBZ.txt"
                        ENDIF
                        IF (choice == 2) THEN
```

```
                    RETURN "WSL.txt"
                ENDIF
                RETURN "WAR.txt"
            ELSE
                IF (choice == 0) OR (choice > 3) THEN
                    Display "Please choose a number from 1 to 3 only."
                    CONTINUE
                ELSE
                    exitingProgram(1)
                ENDIF
            ENDIF
        ENDWHILE
ENDFUNCTION

FUNCTION printResults1(DECLARE VARIABLE partID, model, partName, supID as char
pointer, partQuantity as int)
        Display newline
        printingSeparator(69, '-')
        Display newline
        messageCentered("Inventory Creation Details")
        Display newline
        printingSeparator(69, '-')
        Display "You have successfully registered", partID, “.”
        Display “Model Chosen =", model
        Display “Part ID =", partID
        Display "Part Name =", partName
        Display “Part Quantity =", partQuantity
        Display “Supplier ID =", supID
        Display newline
        printingSeparator(69, '-')
        Display newline
        Display "Details are successfully printed."
        Display newline
```

```
        RETURN
ENDFUNCTION

FUNCTION entryCheckForExistenceSupID (DECLARE VARIABLE entrySupID as char pointer)
        DECLARE VARIABLE flag as int
        flag = 0
        DECLARE VARIABLE fptr as file pointer
        OPEN "supplier.txt" with READ mode AS fptr
        WHILE true
                Display "Please insert your Supplier ID. Enter 'X' to quit:”
                Read entrySupID
                Change entrySupID to uppercase
                seekToNextLine()
                IF entrySupID is X THEN
                        exitingProgram(1)
                ELSE
                        IF length of entrySupID > SUP_ID_SIZE THEN
                                Display “Please input a Supplier ID within”, SUP_ID_SIZE,
                                “characters."
                                CONTINUE
                        ELSE
                                IF (lineCount("supplier.txt") > 0) THEN
                                        flag = entryExists(entrySupID, fptr)
                                        CLOSE fptr
                                        IF (flag == 1) THEN
                                                BREAK
                                        ELSE
                                                exitingProgram(2)
                                        ENDIF
                                ELSE
                                        exitingProgram(6)
                                ENDIF
                        ENDIF
```

```
            ENDIF
        ENDWHILE
        RETURN
ENDFUNCTION

FUNCTION distinctSupIDLineCount()
        DECLARE VARIABLE filePtr as file pointer
        OPEN "supplier.txt" with READ mode AS filePtr
                DECLARE VARIABLE buffer[MAX_PER_LINE_SIZE], supID[SUP_ID_SIZE]
                as char
                DECLARE VARIABLE supFinder as int
                supFinder = 0
                LOOP (DECLARE VARIABLE i as int) FROM 1 TO <= lineCount("supplier.txt")
                STEP 1
                        WHILE Read buffer from filePtr != NULL
                                supID[0] = '\0'
                                Concatenate "SUP" to supID
                                DECLARE VARIABLE
                                placeholderNum[MAX_SIZE_OF_SUPPLIER_NO]  as char
                                placeholderNum = {0}
                                Print i to placeholderNum
                                Concatenate placeholderNum to supID
                                IF buffer contains supID THEN
                                        supFinder = supFinder + 1
                                        BREAK
                                ENDIF
                        ENDWHILE
                        NEXT i
                ENDLOOP
        CLOSE filePtr
        RETURN supFinder
ENDFUNCTION
```

```
FUNCTION partsInventoryCreation()
        Assigning part to warehouse STRUCTURE
        DECLARE VARIABLE fileName as char pointer
        fileName = selectingWarehouse()
        DECLARE VARIABLE model[MODEL_SIZE] as char
        IF fileName is "WBZ.txt" THEN
                Copy "Blaze" to model
                Copy "WBZ" to part.warehouseChoice
        ENDIF
        IF fileName is "WSL.txt" THEN
                Copy "Silk" to model
                Copy "WSL" to part.warehouseChoice
        ENDIF
        IF fileName is "WAR.txt" THEN
                Copy "Armer" to model
                Copy "WAR" to part.warehouseChoice
        ENDIF
        printingSeparator(69, '-')
        Display newline
        part.partName[0] = '\0'
        entryCheckForDuplicate(fileName, "part", PART_NAME_SIZE, part.partName)
        Change part.partName to uppercase
        printingSeparator(69, '-')
        Display newline
        part.partQuantity = '\0'
        Display "Please insert the quantity of part (‘, part.partName, ‘) that you would like to
        register"
        Display "Enter negative numbers to quit:"
        Read part.partQuantity
        IF (part.partQuantity < 0) THEN
                exitingProgram(1)
        ENDIF
        printingSeparator(69, '-')
```

```
Display newline
DECLARE VARIABLE counterForPart as int
counterForPart = 0
counterForPart = lineCount("WBZ.txt") + lineCount("WSL.txt") + lineCount("WAR.txt")
part.partID[0] = '\0’
DECLARE VARIABLE placeholderNum[MAX_SIZE_OF_PART_NO] as char
placeholderNum[MAX_SIZE_OF_PART_NO] = {0}
Concatenate part.warehouseChoice to part.partID
Print counterForPart + 1 to placeholderNum
Concatenate placeholderNum to part.partID
Assigning sup to supplier STRUCTURE
DECLARE VARIABLE option as char
Display "Do you have an existing Supplier ID? [ Y (Yes) / N (No) / X (To quit) ]”
Read option
Change option to uppercase
seekToNextLine()
IF (option == 'X') THEN
    exitingProgram(1)
ELSE
    IF (option == 'Y') THEN
        IF (lineCount("supplier.txt") == 0) THEN
            exitingProgram(6)
        ELSE
            entryCheckForExistenceSupID(sup.supID)
            DECLARE VARIABLE fSection1PartExtra as file pointer
            OPEN fileName with APPEND mode AS fSection1PartExtra
                IF (part.partQuantity < 10) THEN
                    Write formatted output to fSection1PartExtra
                    "%s|%s|%d|%s|LOW\n", part.partID, part.partName,
                    part.partQuantity, sup.supID
                ELSE
```

```
                                        Write formatted output to fSection1PartExtra
                                        "%s|%s|%d|%s|NORMAL\n", part.partID,
                                        part.partName, part.partQuantity, sup.supID
                                    ENDIF
                                CLOSE fSection1PartExtra
                                lineNumberFinderAndPrintResults("supplier.txt", sup.supID, 4,
                                part.partID)
                                printResults1(part.partID, model, part.partName, part.partQuantity,
                                sup.supID)
                                exitingProgram(1)
                        ENDIF
                ELSE
                        IF (option == 'N') THEN
                                entryCheckForDuplicate("supplier.txt", "name",
                                SUP_NAME_SIZE, sup.supName)
                                Change sup.supName to uppercase
                                DECLARE VARIABLE counterForSupplier as int
                                counterForSupplier = distinctSupIDLineCount()
                                sup.supID[0] = '\0'
                                DECLARE VARIABLE
                                placeholderNum2[MAX_SIZE_OF_SUPPLIER_NO] as char
                                placeholderNum2[MAX_SIZE_OF_SUPPLIER_NO] = {0}
                                Concatenate "SUP" to sup.supID
                                Print counterForSupplier + 1 to placeholderNum2
                                Concatenate placeholderNum2 to sup.supID
                        ELSE
                                exitingProgram(3)
                        ENDIF
                ENDIF
        ENDIF
        DECLARE VARIABLE fSection1Part as file pointer
        OPEN fileName with APPEND mode AS fSection1Part
                IF (part.partQuantity < 10) THEN
```

```
                Write formatted output to fSection1Part "%s|%s|%d|%s|LOW\n",
                part.partID, part.partName, part.partQuantity, sup.supID
            ELSE
                Write formatted output to fSection1Part "%s|%s|%d|%s|NORMAL\n",
                part.partID, part.partName, part.partQuantity, sup.supID
            ENDIF
        CLOSE fSection1Part
        DECLARE VARIABLE fSection1Part as file pointer
        OPEN "supplier.txt" with APPEND mode AS fSection1Sup
            Write formatted output to fSection1Sup "%s|%s|%s\n", sup.supID, sup.supName,
            part.partID
        CLOSE fSection1Sup
        printResults1(part.partID, model, part.partName, part.partQuantity, sup.supID)
        RETURN
ENDFUNCTION

FUNCTION partsInventoryUpdate()
        Assigning part to warehouse STRUCTURE
        DECLARE VARIABLE fileName as char pointer
        fileName = entryCheckForExistencePartID(part.partID)
        lineNumberFinderAndPrintResults(fileName, part.partID, 3, "Empty")
        RETURN
ENDFUNCTION

FUNCTION partsInventoryTracking()
        DECLARE VARIABLE fileName as char pointer
        fileName = selectingWarehouse()
        DECLARE VARIABLE filePtr as file pointer
        DECLARE VARIABLE storeLine[MAX_PER_LINE_SIZE] as char
        OPEN fileName with READ mode AS filePtr
        DECLARE VARIABLE LOWStatusCounter as int
        LOWStatusCounter = 0
        DECLARE VARIABLE choice as int
```

```
printingSeparator(69, '-')
Display “Pick an operation by entering the corresponding number:"
Display “1 - Print all parts in ascending order (According to Part ID)”
Display “2 - Print parts that has < 10 stock"
WHILE true
        Display "Enter negative numbers to quit:"
        Read choice
        seekToNextLine()
        IF (lineCount(fileName) == 0) THEN
                Display "This file has no records yet."
                BREAK
        ENDIF
        IF (choice > 0) AND (choice < 3) THEN
                Display newline
                printingSeparator(69, '-')
                Display newline
                messageCentered("Parts Inventory Details")
                Display newline
                printingSeparator(69, '-')
                Display newline
                IF (choice == 1) THEN
                        WHILE Read storeLine from filePtr != NULL
                                Display storeLine
                        ENDWHILE
                ELSE
                        WHILE Read storeLine from filePtr != NULL
                                IF storeLine contains “LOW” THEN
                                        LOWStatusCounter = LOWStatusCounter + 1
                                        Display storeLine
                                ENDIF
                        ENDWHILE
                        IF (LOWStatusCounter == 0) THEN
                                Display "No parts are less than 10 in”, fileName, “.”
```

```
                    ENDIF
                ENDIF
                CLOSE filePtr
                BREAK
            ELSE
                IF (choice == 0) OR (choice > 2) THEN
                    Display "Please choose a number from 1 to 2 only."
                ELSE
                    exitingProgram(1)
                ENDIF
            ENDIF
        ENDWHILE
        RETURN
ENDFUNCTION

FUNCTION searchRecordsAndSupplierDetails()
        Assigning part to warehouse STRUCTURE
        DECLARE VARIABLE fileName as char pointer
        fileName = entryCheckForExistencePartID(part.partID)
        DECLARE VARIABLE choice as int
        printingSeparator(69, '-')
        Display "Pick an operation by entering the corresponding number:"
        Display "1 - View Part Details”
        Display “2 - View Supplier Details"
        WHILE true
            Display "Enter negative numbers to quit:"
            Read choice
            seekToNextLine()
            IF (choice > 0) AND (choice < 3) THEN
                IF (choice == 1) THEN
                    lineNumberFinderAndPrintResults(fileName, part.partID, 1,
                    "Empty")
                ELSE
```

```
                    lineNumberFinderAndPrintResults("supplier.txt", part.partID, 2,
                    "Empty")
                ENDIF
                BREAK
            ELSE
                IF (choice == 0) OR (choice > 2) THEN
                    Display "Please choose a number from 1 to 2 only."
                ELSE
                    exitingProgram(1)
                ENDIF
            ENDIF
        ENDWHILE
        RETURN
ENDFUNCTION

FUNCTION fileCheck()
        DECLARE VARIABLE fptr as file pointer
        DECLARE VARIABLE files[4][FILENAME_MAX] as char
        files[4][FILENAME_MAX] = {"WBZ.txt", "WSL.txt", "WAR.txt", "supplier.txt"}
        DECLARE VARIABLE i as int
        LOOP (DECLARE i as int) FROM 0 TO 3 STEP 1
            IF files doesn't exist THEN
                OPEN files[i] with WRITE mode AS fptr
                CLOSE fptr
            ENDIF
            NEXT i
        ENDLOOP
        RETURN
ENDFUNCTION

FUNCTION printMenu()
        Display newline
        printingSeparator(69, '=')
```

```
        Display newline
        messageCentered("Welcome to the Automobile Parts Management System")
        Display newline
        printingSeparator(69, '=')
        Display “1. Parts Inventory Creation In Warehouse”
        Display “2. Parts Inventory Update”
        Display “3. Parts Inventory Tracking"
        Display “4. Search Records & Supplier Details”
        Display “5. Exit"
        RETURN
ENDFUNCTION

FUNCTION menuSelection()
        DECLARE VARIABLE choice as int
        WHILE true
                fileCheck()
                printMenu()
                Display "Please choose any operation from the given options:"
                Read choice
                seekToNextLine()
                WHILE while (choice <= 0) OR (choice > 5)
                        Display "Please choose a number from 1 to 5 only."
                        Display “Please choose any operation from the given options:"
                        Read choice
                        seekToNextLine()
                ENDWHILE
                CASE OF choice
                        = 1 :
                        printingSeparator(69, '=')
                        partsInventoryCreation()
                        BREAK
                        = 2 :
                        printingSeparator(69, '=')
```

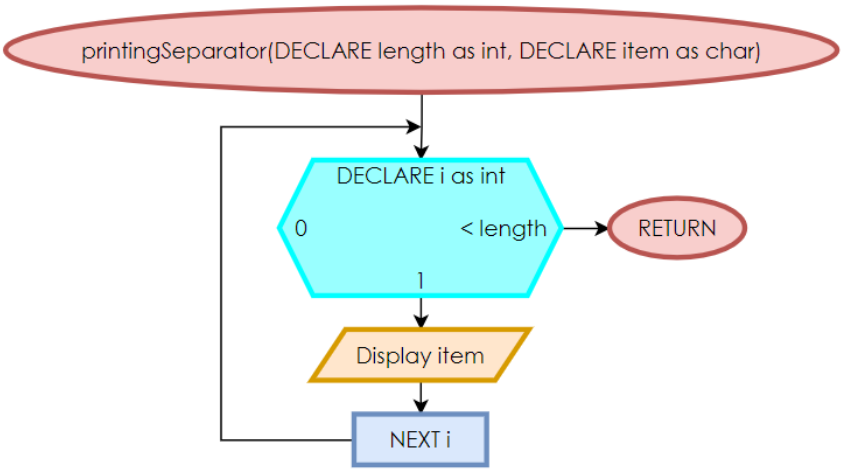
```
                    IF (lineCount("supplier.txt") > 0) THEN
                        partsInventoryUpdate()
                    ELSE
                        Display "No record exists.Please register in 'Section 1 - Parts
                        Inventory Creation In Warehouse' before proceeding to this section."
                    ENDIF
                    BREAK
                    = 3 :
                    printingSeparator(69, '=')
                    IF (lineCount("supplier.txt") > 0) THEN
                        partsInventoryTracking()
                    ELSE
                        Display "No record exists.Please register in 'Section 1 - Parts
                        Inventory Creation In Warehouse' before proceeding to this section."
                    ENDIF
                    BREAK
                    = 4 :
                    printingSeparator(69, '=')
                    IF (lineCount("supplier.txt") > 0) THEN
                        searchRecordsAndSupplierDetails()
                    ELSE
                        Display "No record exists.Please register in 'Section 1 - Parts
                        Inventory Creation In Warehouse' before proceeding to this section."
                    ENDIF
                    BREAK
                OTHERWISE
                    Display "Thank you for using this system :D"
                    printingSeparator(69, '=')
                    Exit program
                ENDCASE
            ENDWHILE
            RETURN
        ENDFUNCTION
```

```
    FUNCTION main()
        menuSelection()
        RETURN 0
    ENDFUNCTION
END
```

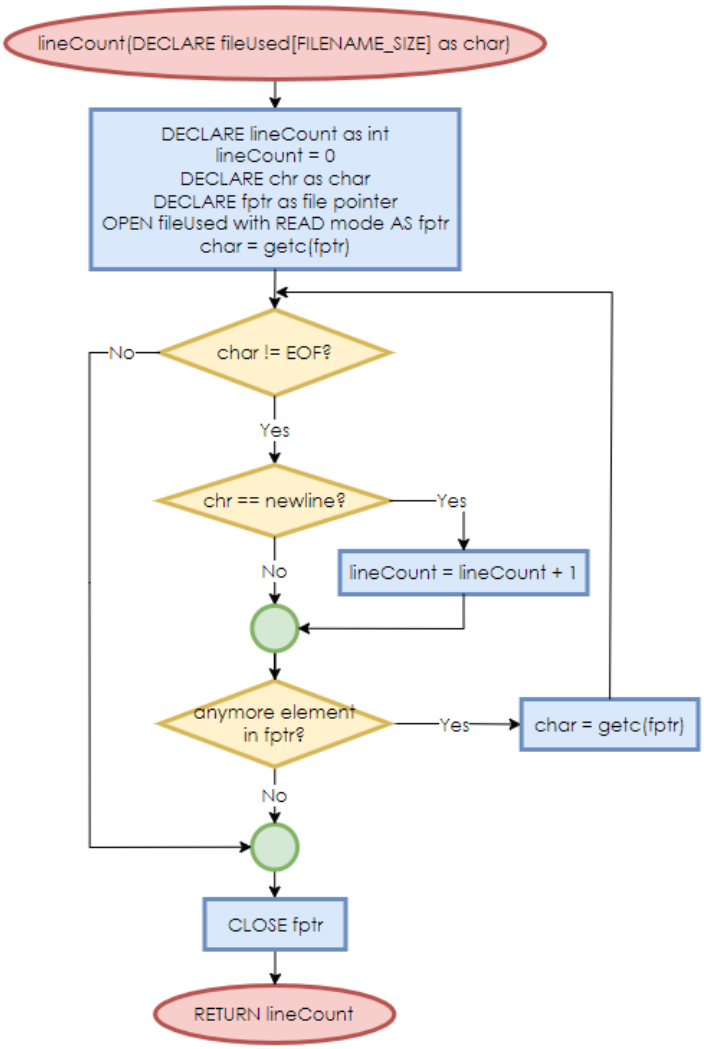
FLOWCHART

GENERAL FUNCTIONS

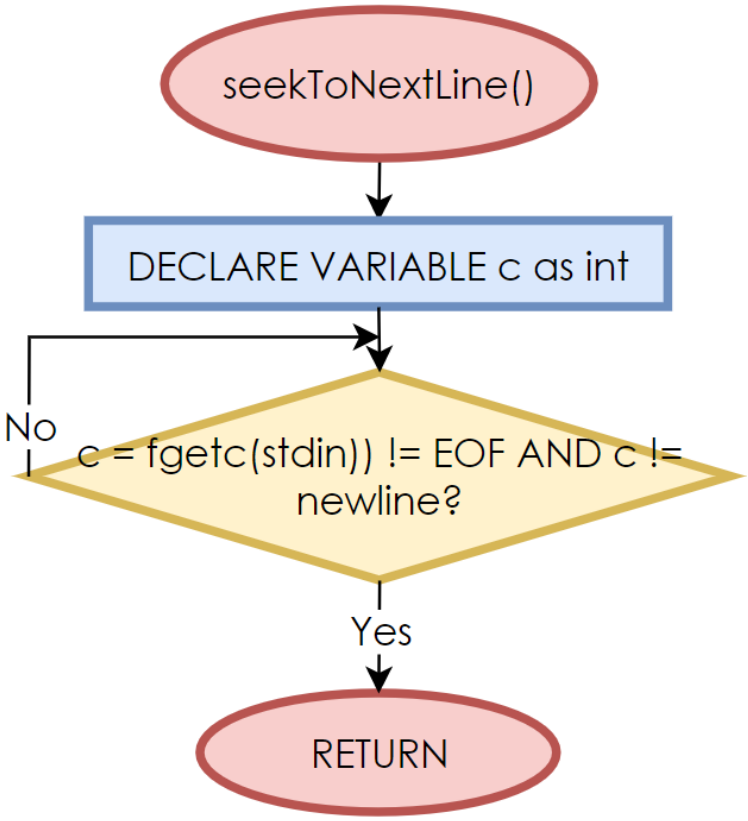
void printingSeparator(int length, char item)



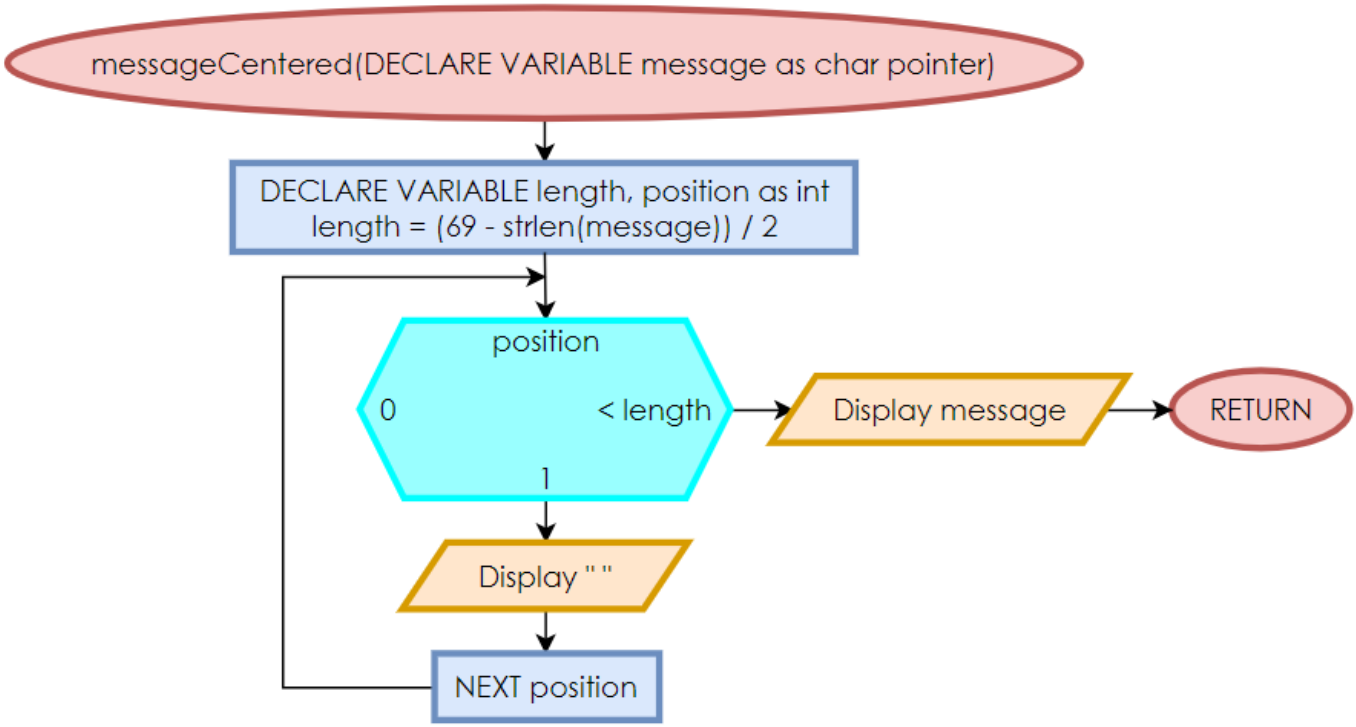
int lineCount(char fileUsed[FILENAME_SIZE])



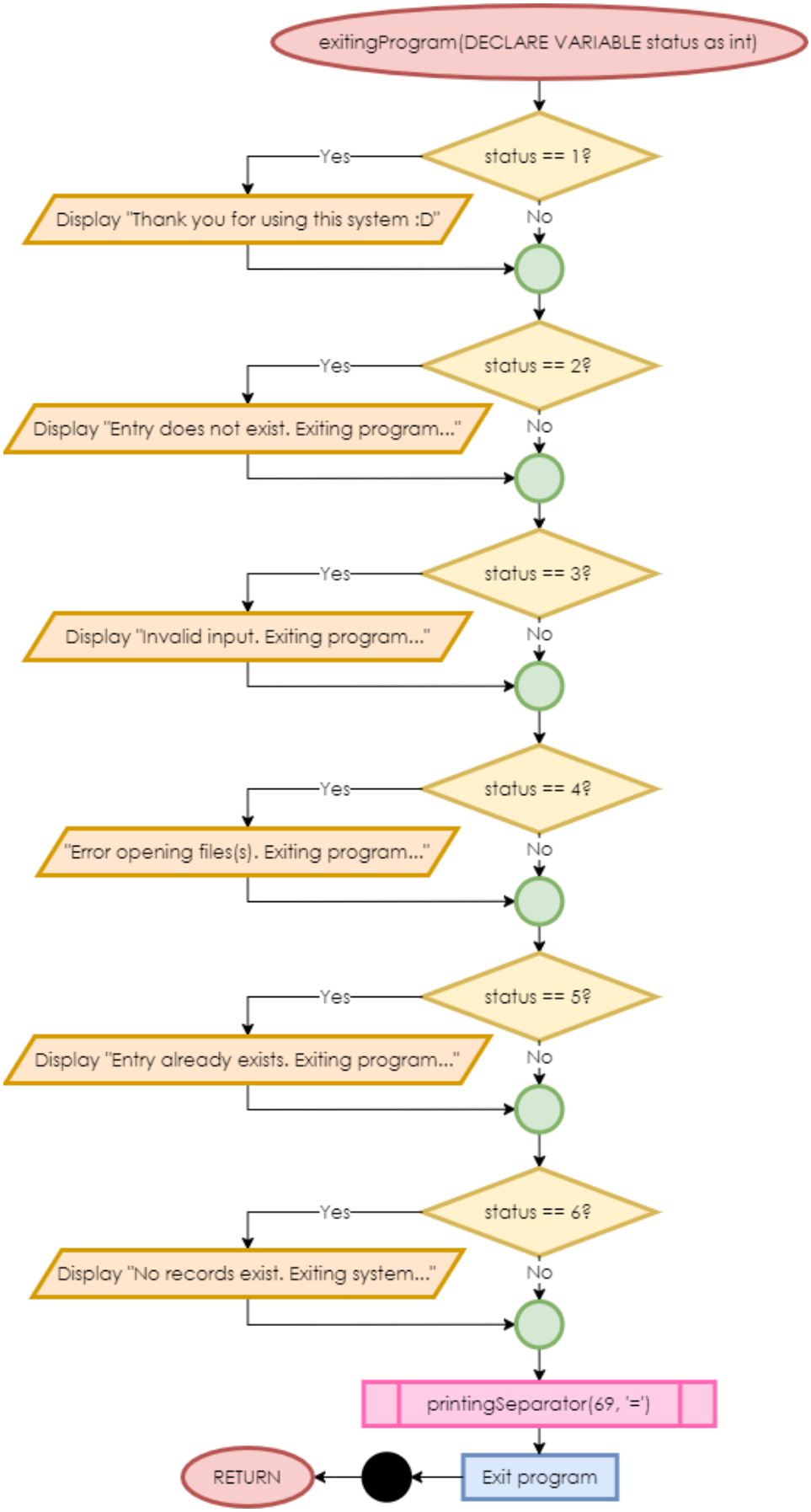
void seekToNextLine()



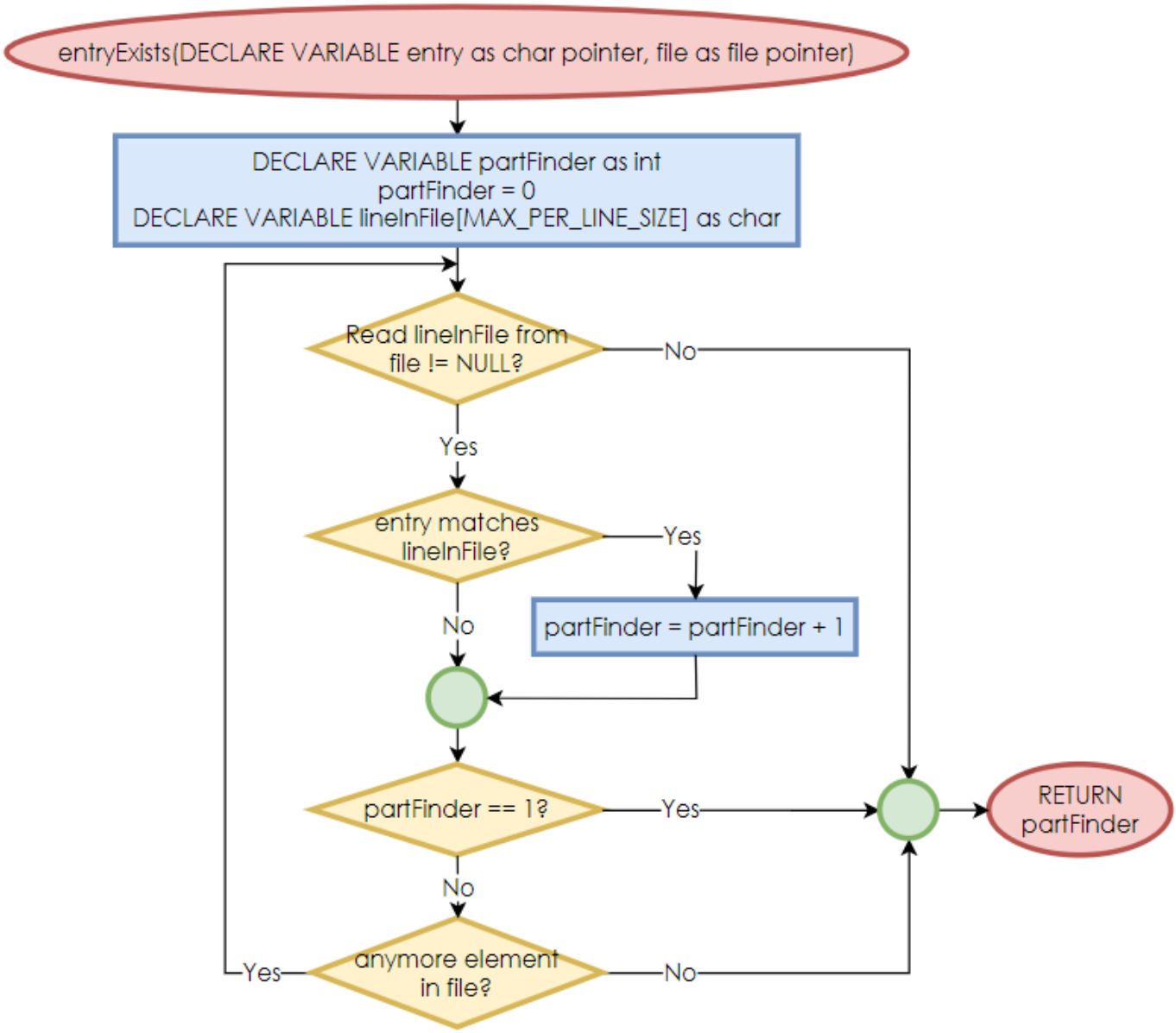
void messageCentered(char *message)



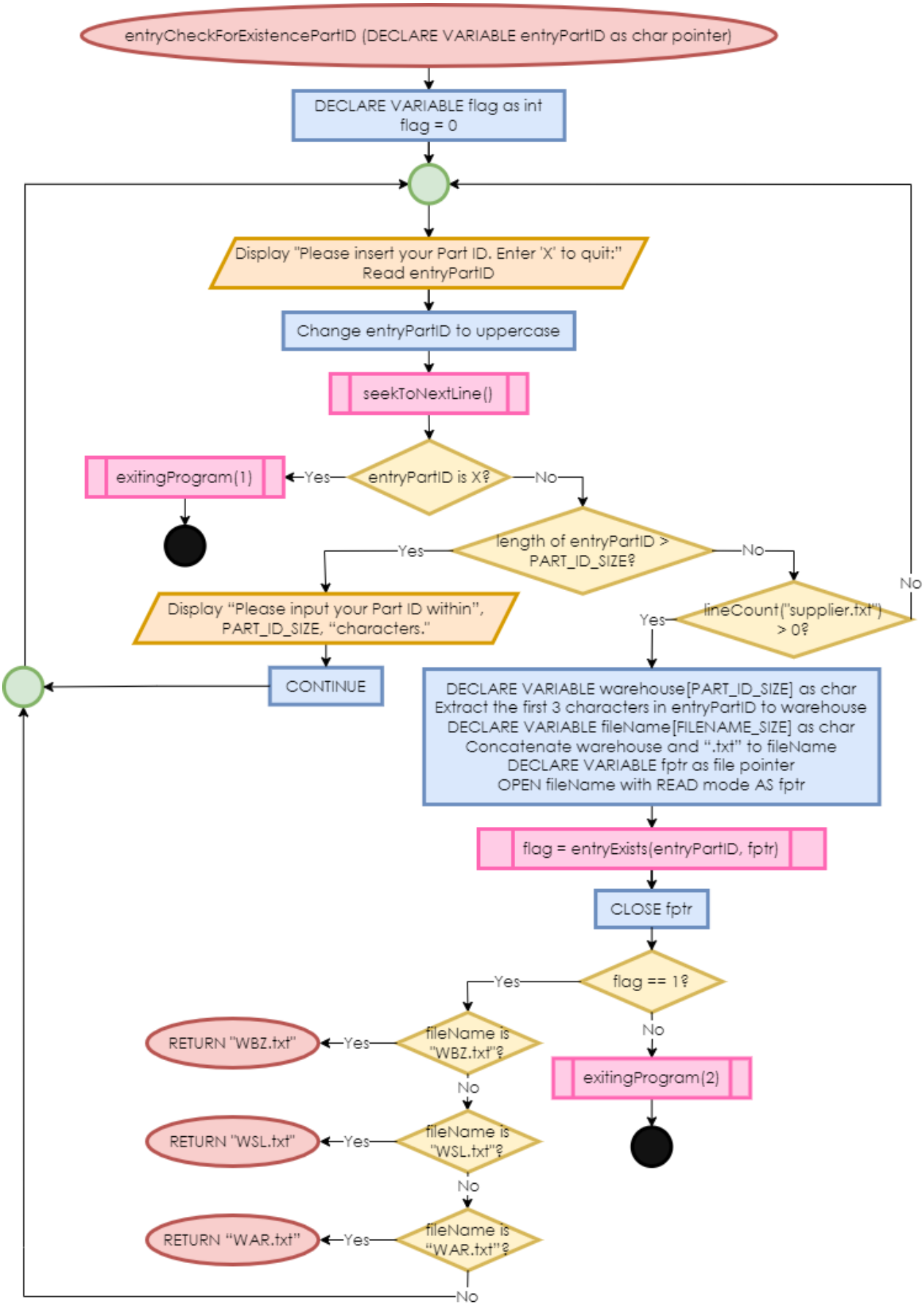
void exitingProgram(int status)



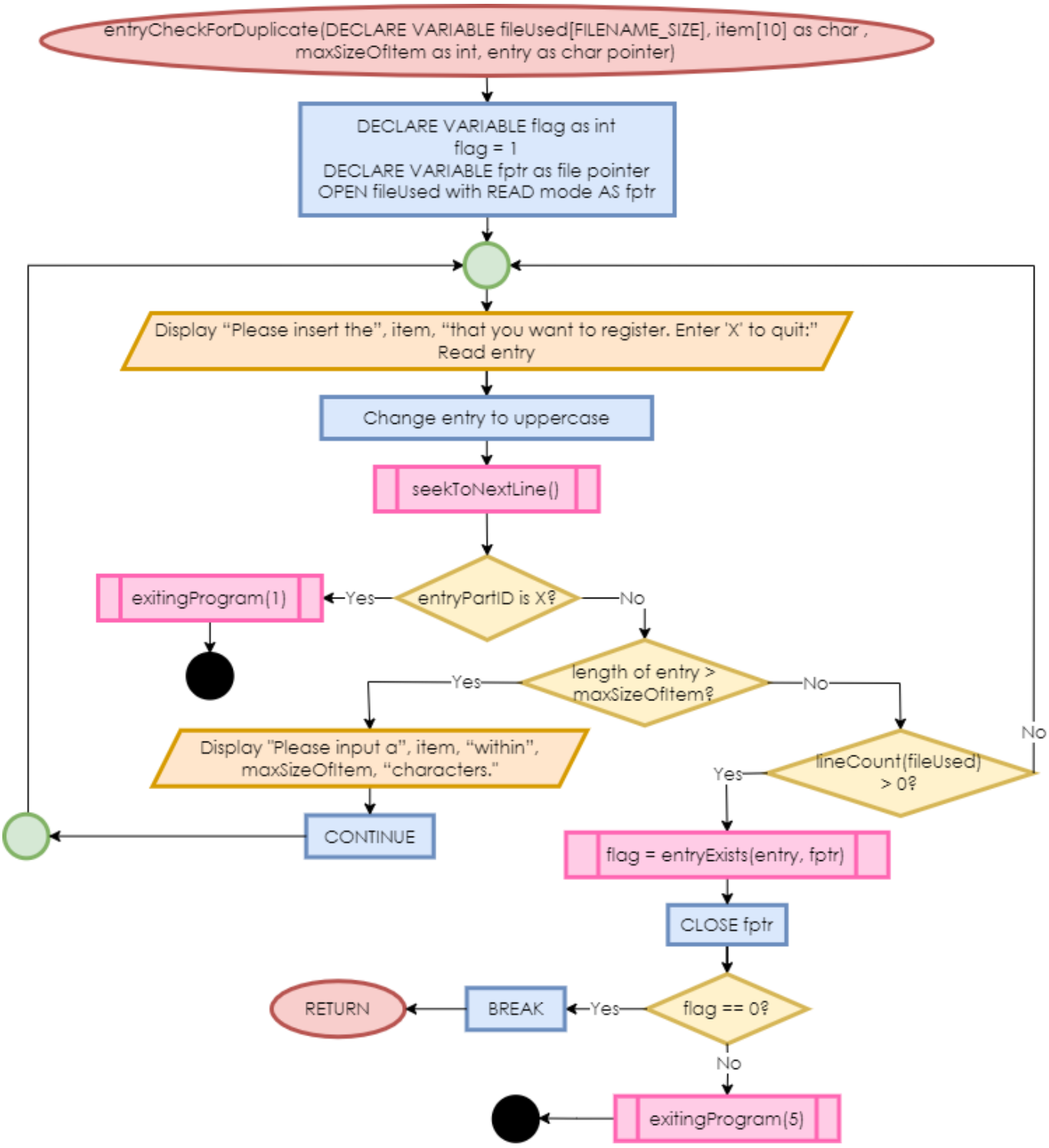
int entryExists(char *entry, FILE *file)



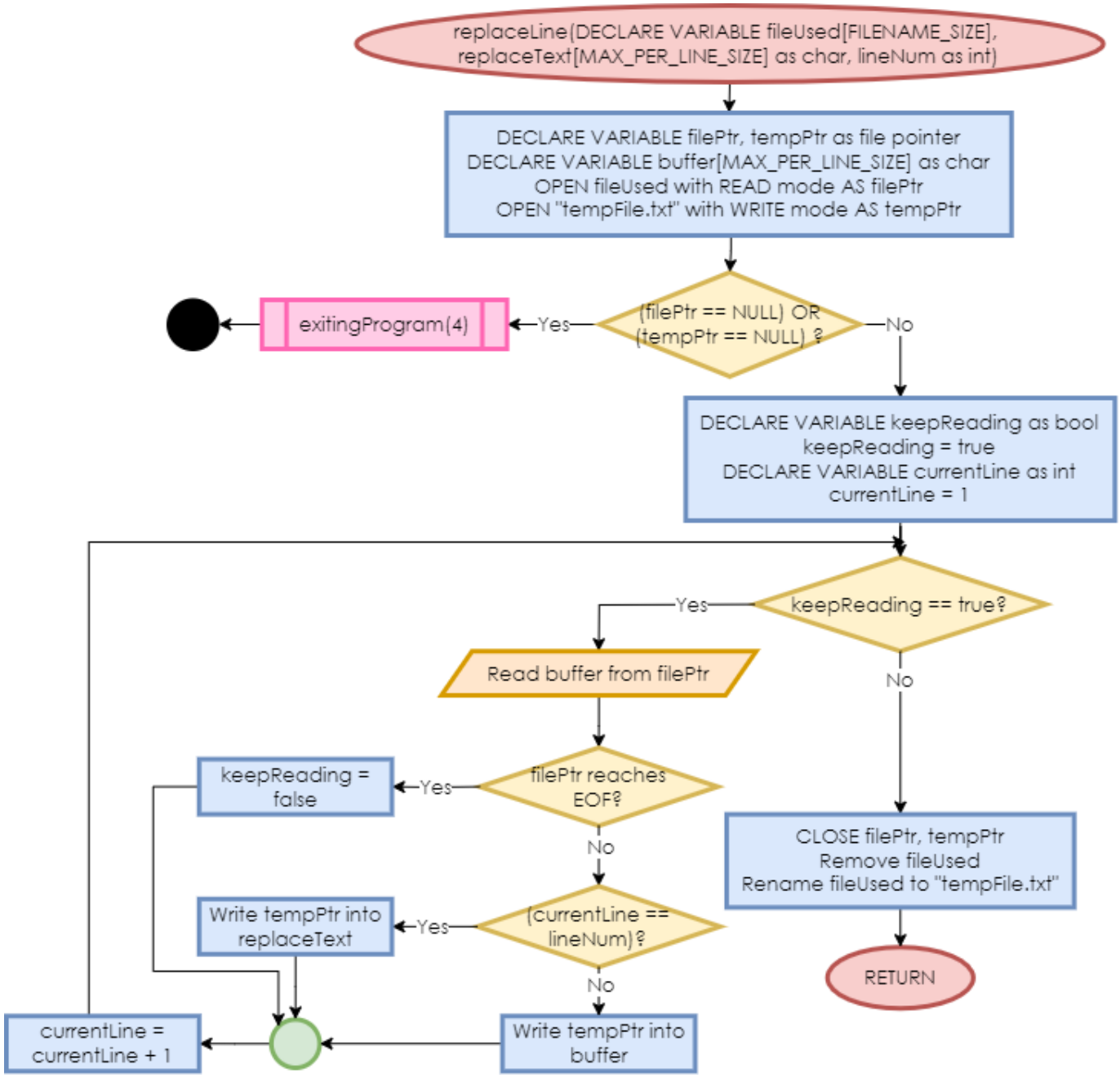
char *entryCheckForExistencePartID(char *entryPartID)



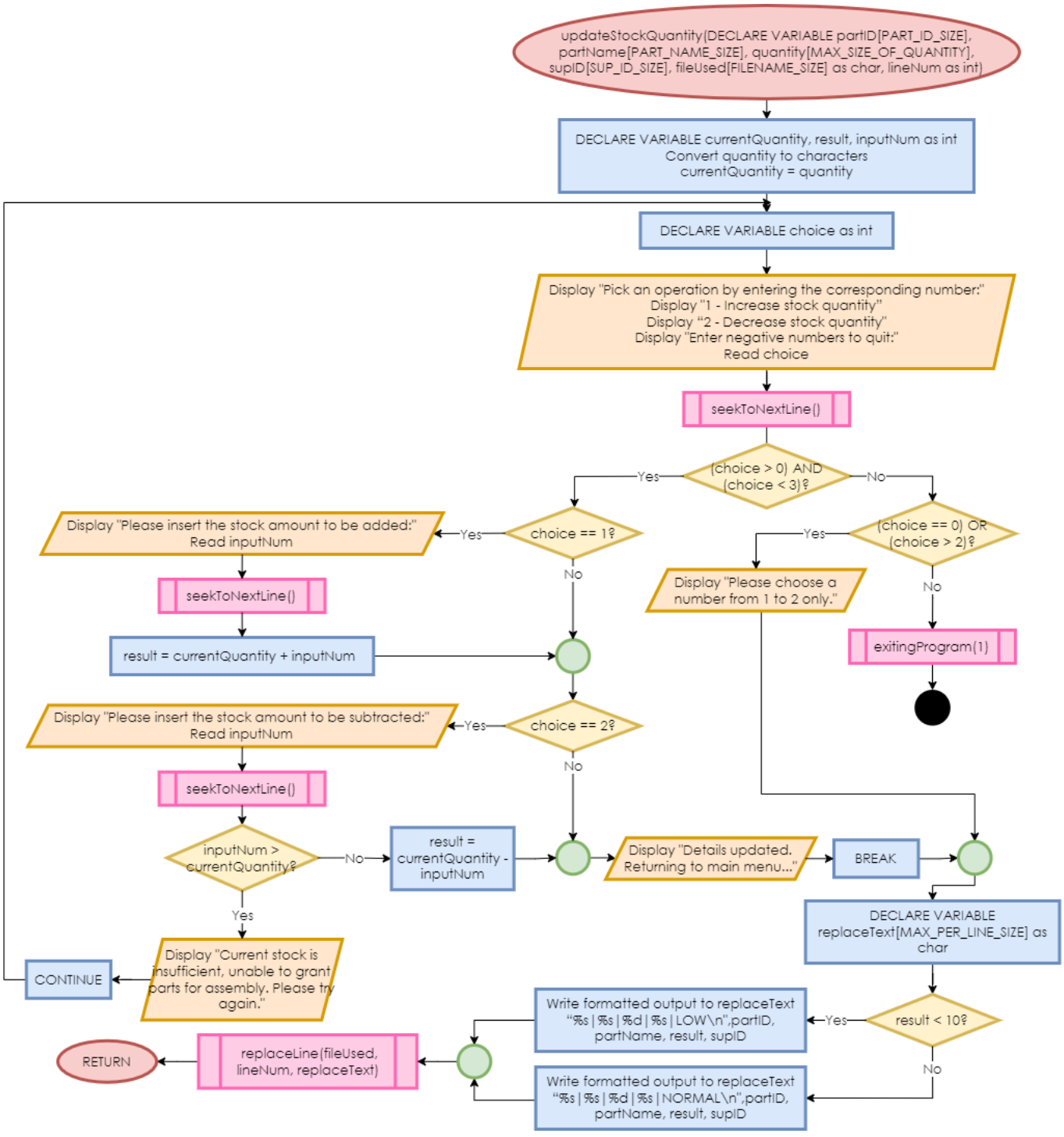
void entryCheckForDuplicate(char fileUsed[FILENAME_SIZE], char item[10], int maxSizeOfItem, char *entry)



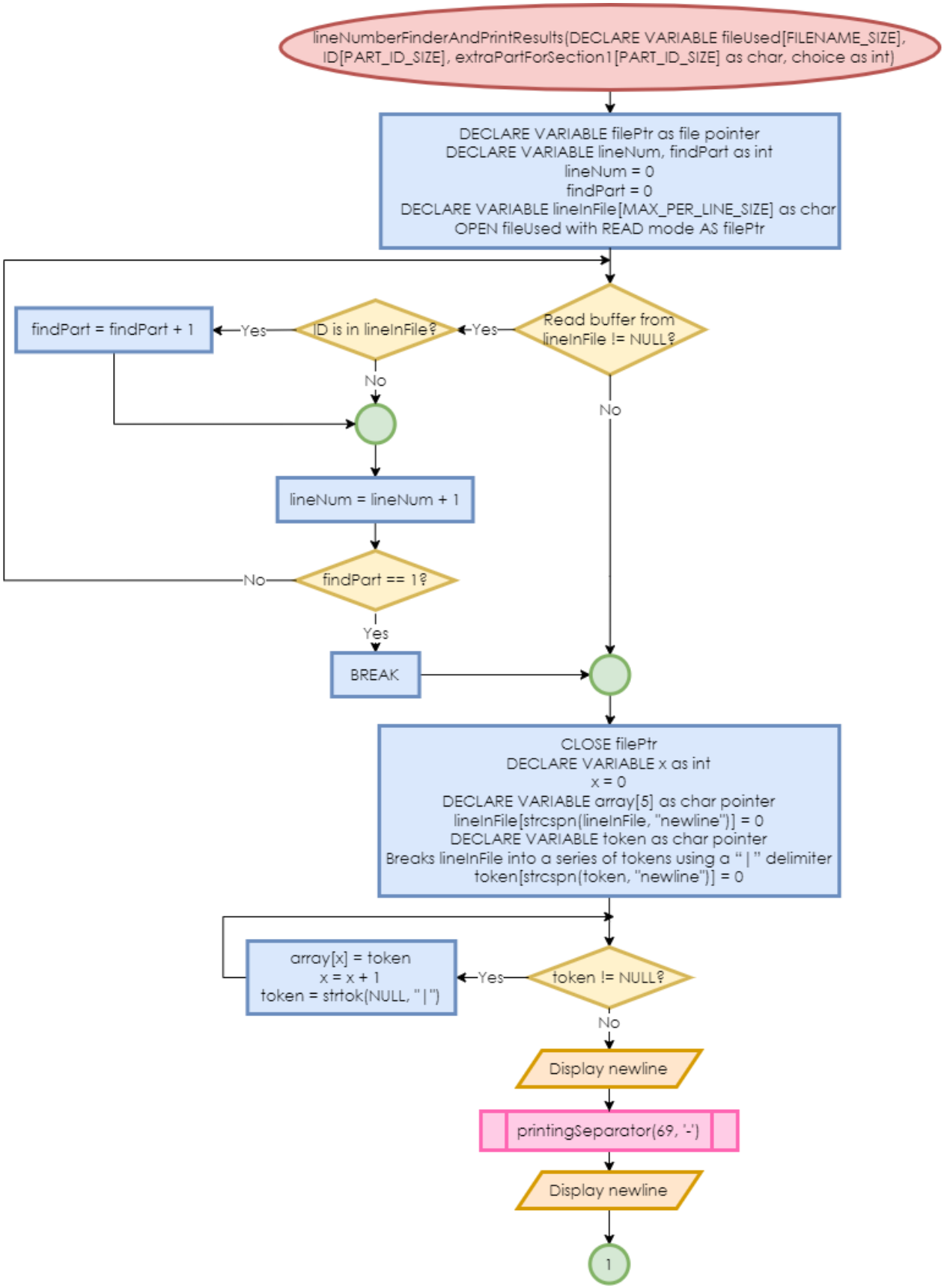
void replaceLine(char fileUsed[FILENAME_SIZE], int lineNum, char replaceText[MAX_PER_LINE_SIZE])

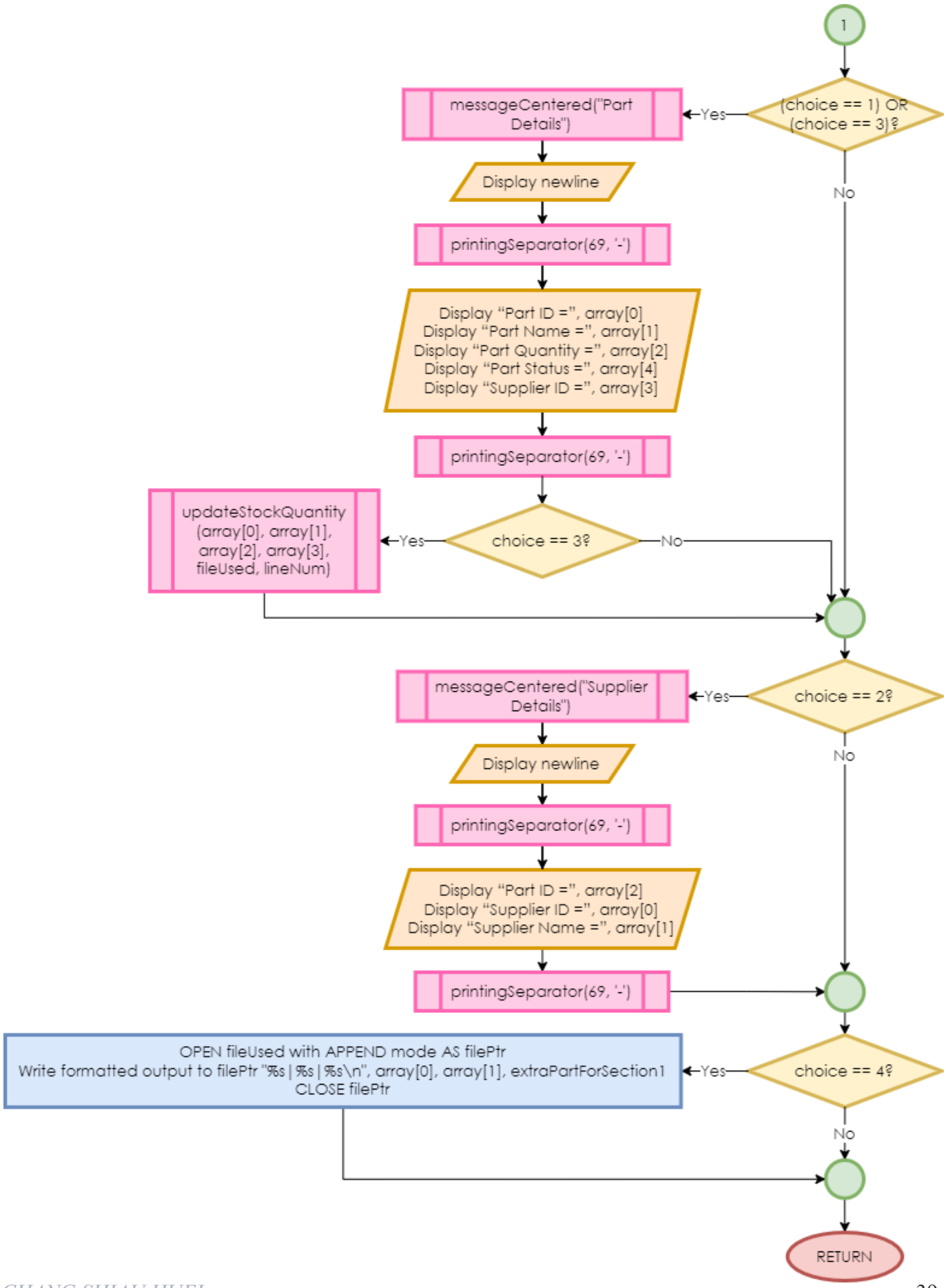


void updateStockQuantity(char partID[PART_ID_SIZE], char partName[PART_NAME_SIZE], char quantity[MAX_SIZE_OF_QUANTITY], char supID[SUP_ID_SIZE], char fileUsed[FILENAME_SIZE], int lineNum)

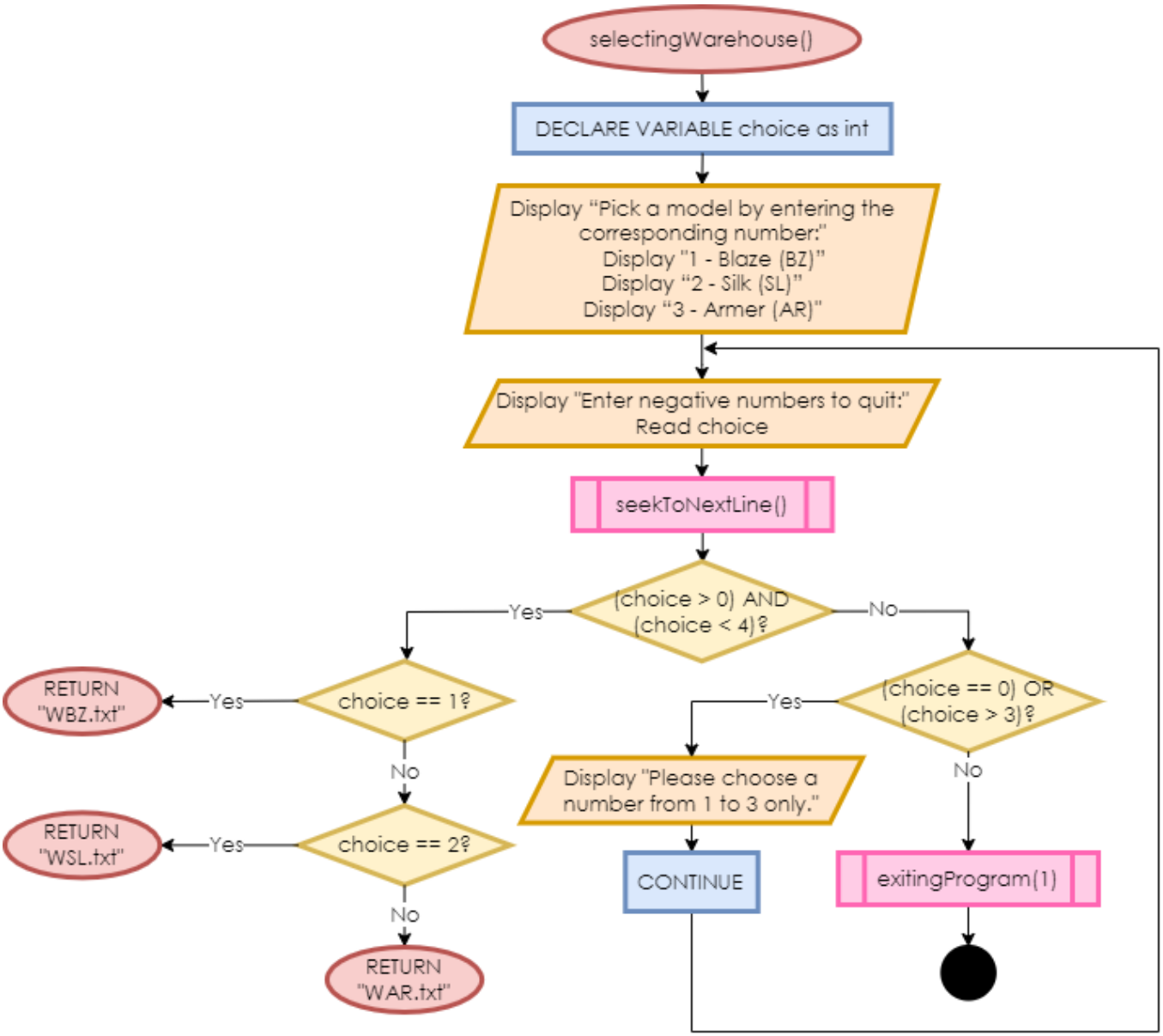


void lineNumberFinderAndPrintResults(char fileUsed[FILENAME_SIZE], char ID[PART_ID_SIZE], int choice, char extraPartForSection1[PART_ID_SIZE])



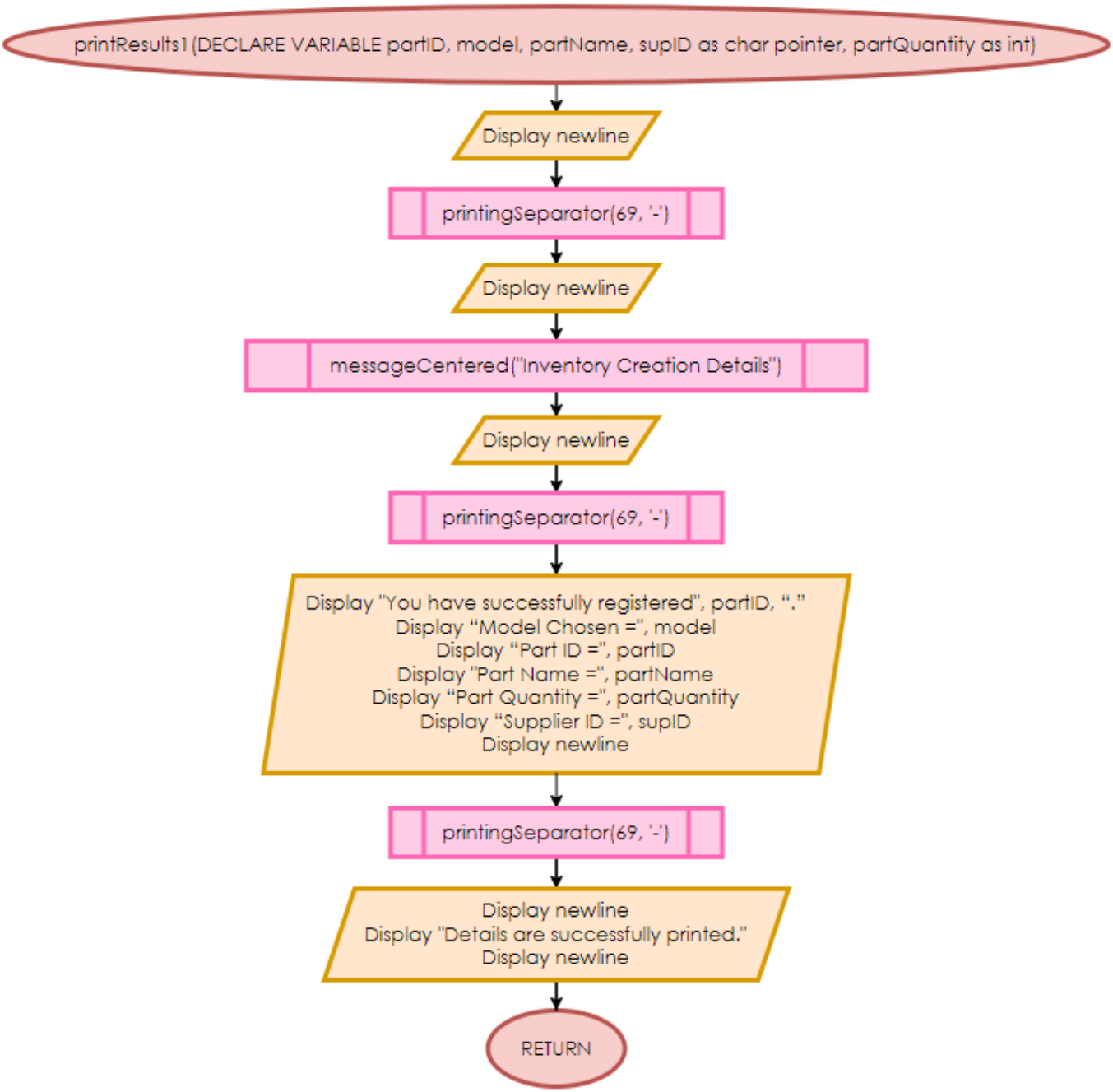

1
(choice == 1) OR (choice == 3)?
Yes
messageCentered("Part Details")
No
Display newline
printingSeparator(69, '-')
Display "Part ID =", array[0]
Display "Part Name =", array[1]
Display "Part Quantity =", array[2]
Display "Part Status =", array[4]
Display "Supplier ID =", array[3]
printingSeparator(69, '-')
choice == 3?
Yes
No
updateStockQuantity (array[0], array[1], array[2], array[3], fileUsed, lineNum)
choice == 2?
Yes
messageCentered("Supplier Details")
No
Display newline
printingSeparator(69, '-')
Display "Part ID =", array[2]
Display "Supplier ID =", array[0]
Display "Supplier Name =", array[1]
printingSeparator(69, '-')
choice == 4?
Yes
OPEN fileUsed with APPEND mode AS filePtr
Write formatted output to filePtr "%s|%s|%s\n", array[0], array[1], extraPartForSection1
CLOSE filePtr
No
RETURN

char *selectingWarehouse()

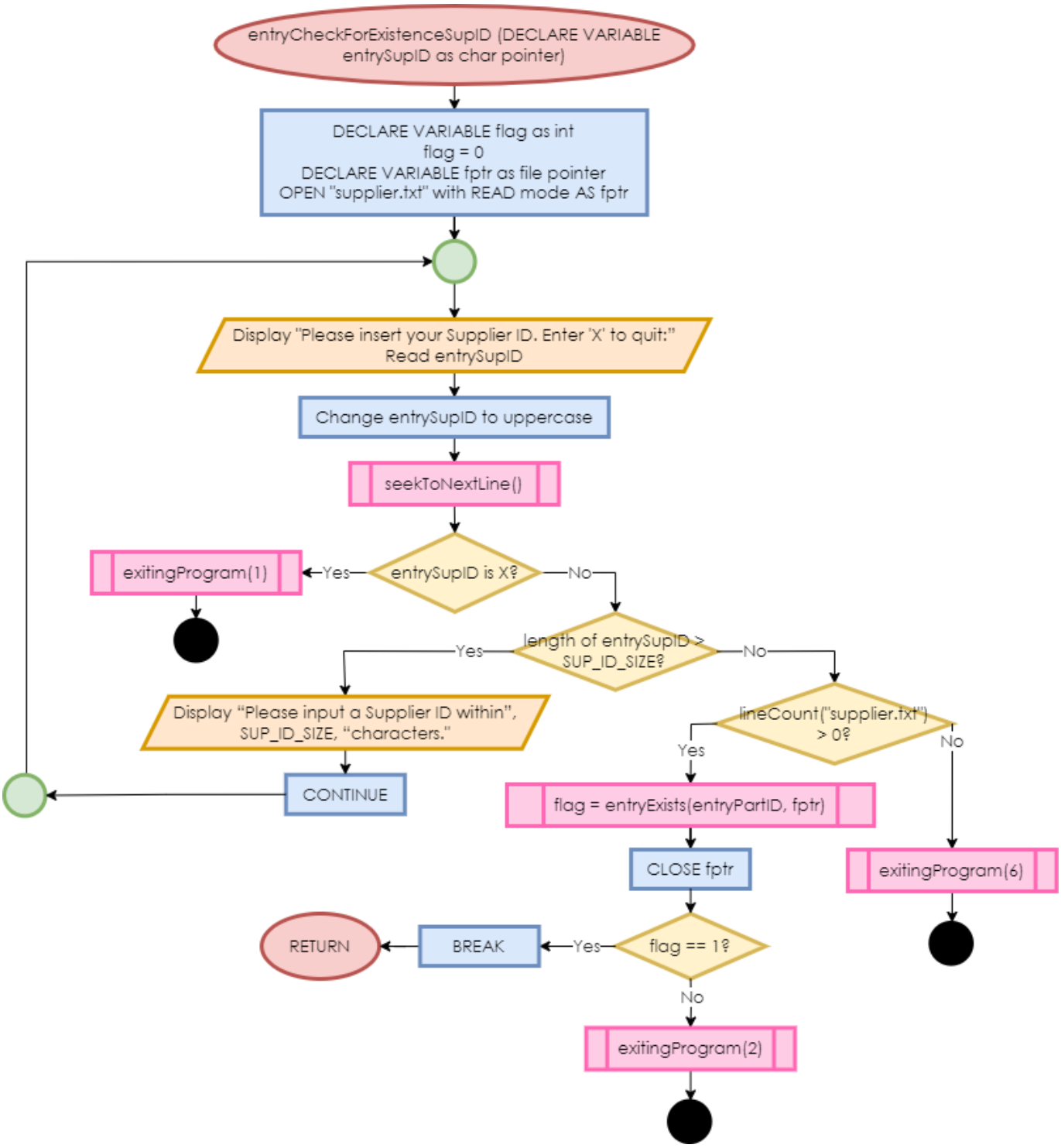


SECTION 1 FUNCTIONS

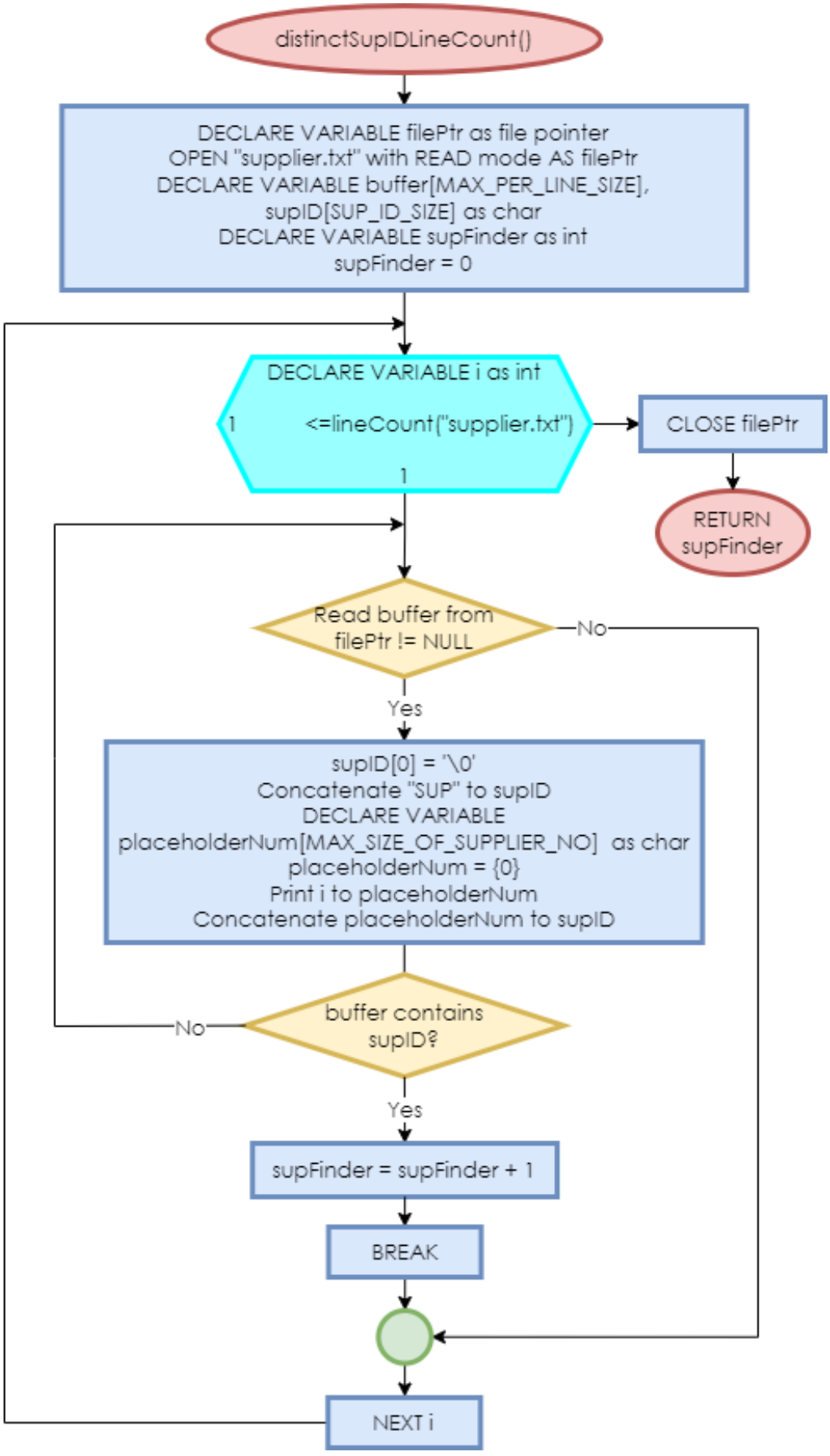
void printResults1(char *partID, char* model, char *partName, int partQuantity, char* supID)



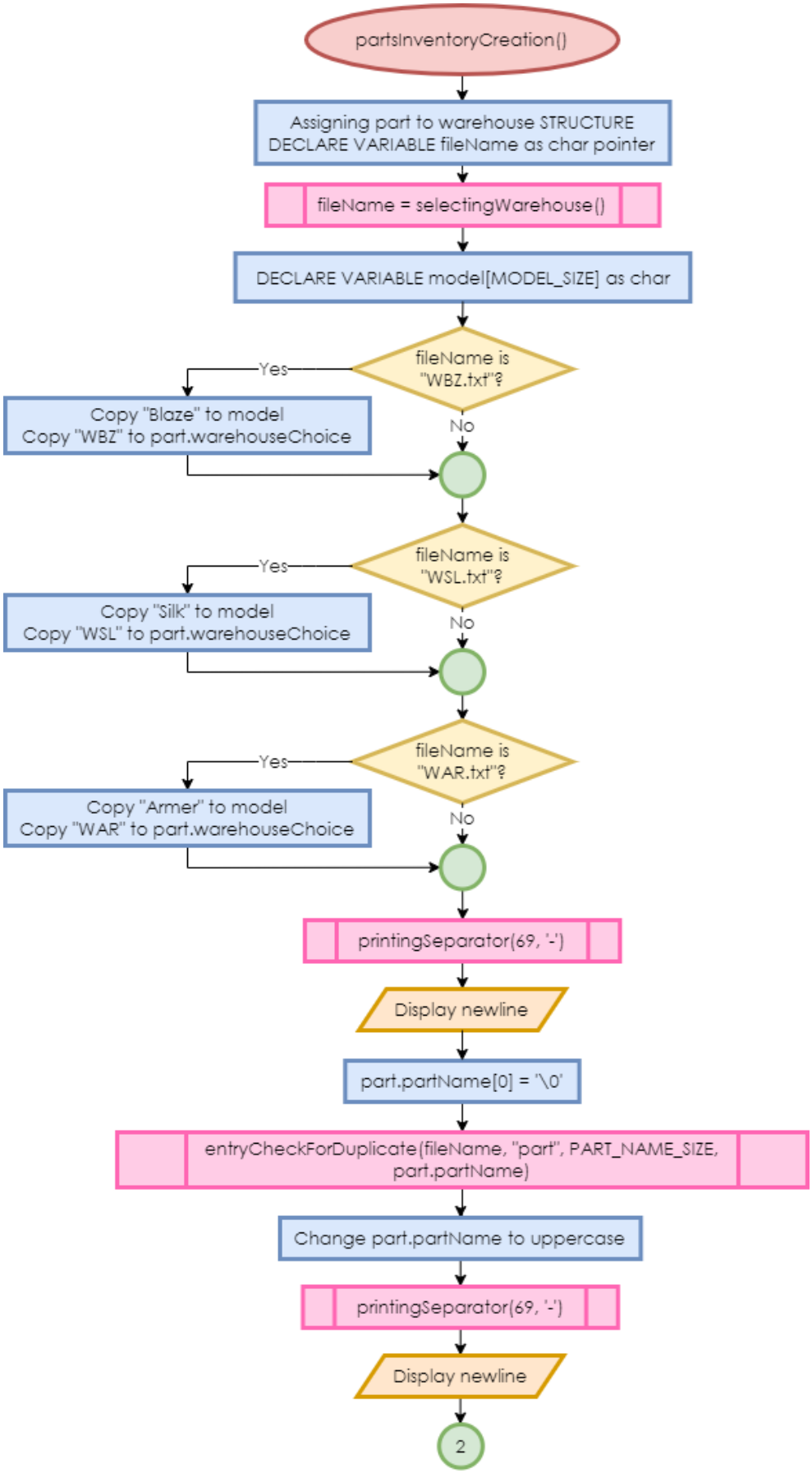
void entryCheckForExistenceSupID(char *entrySupID)

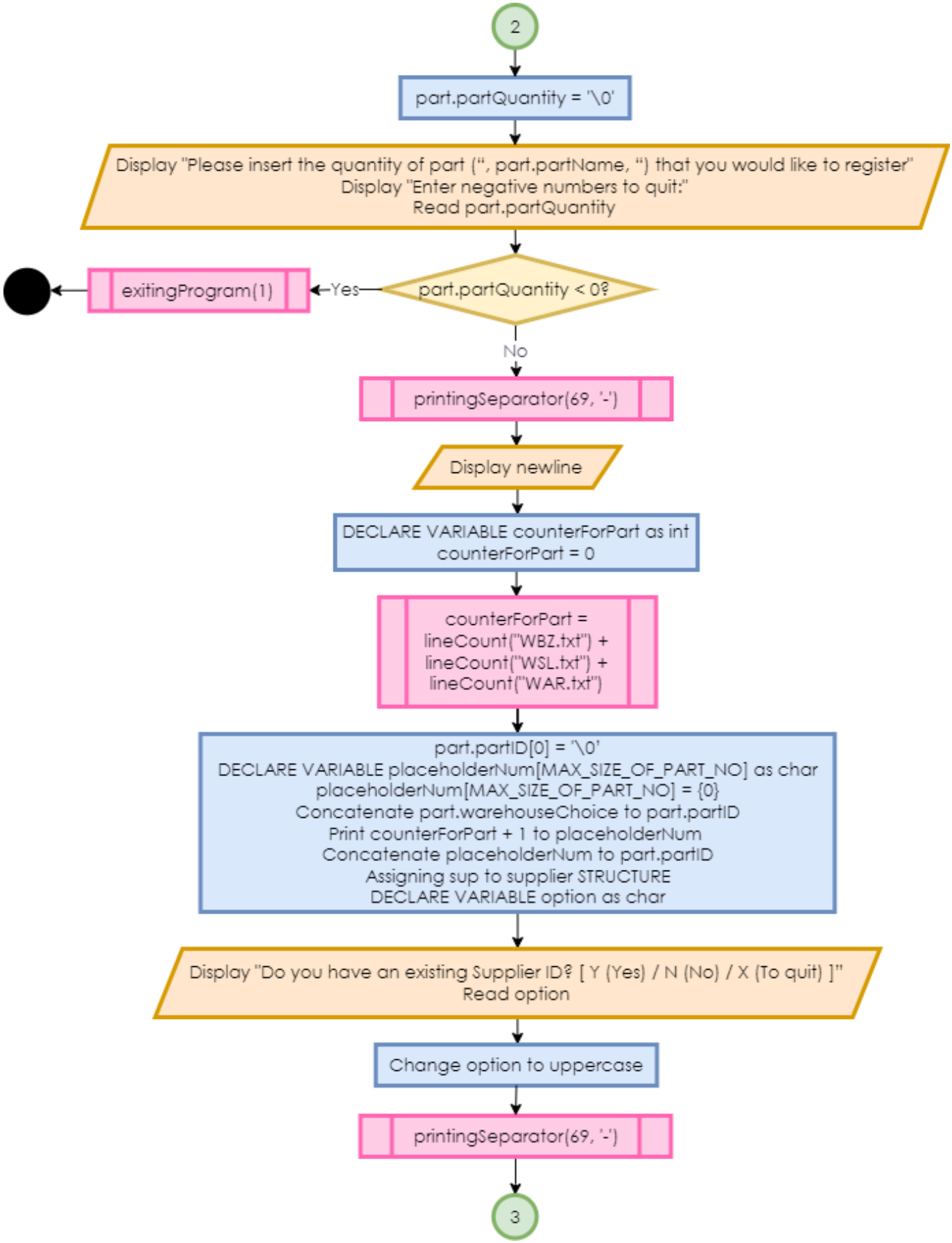


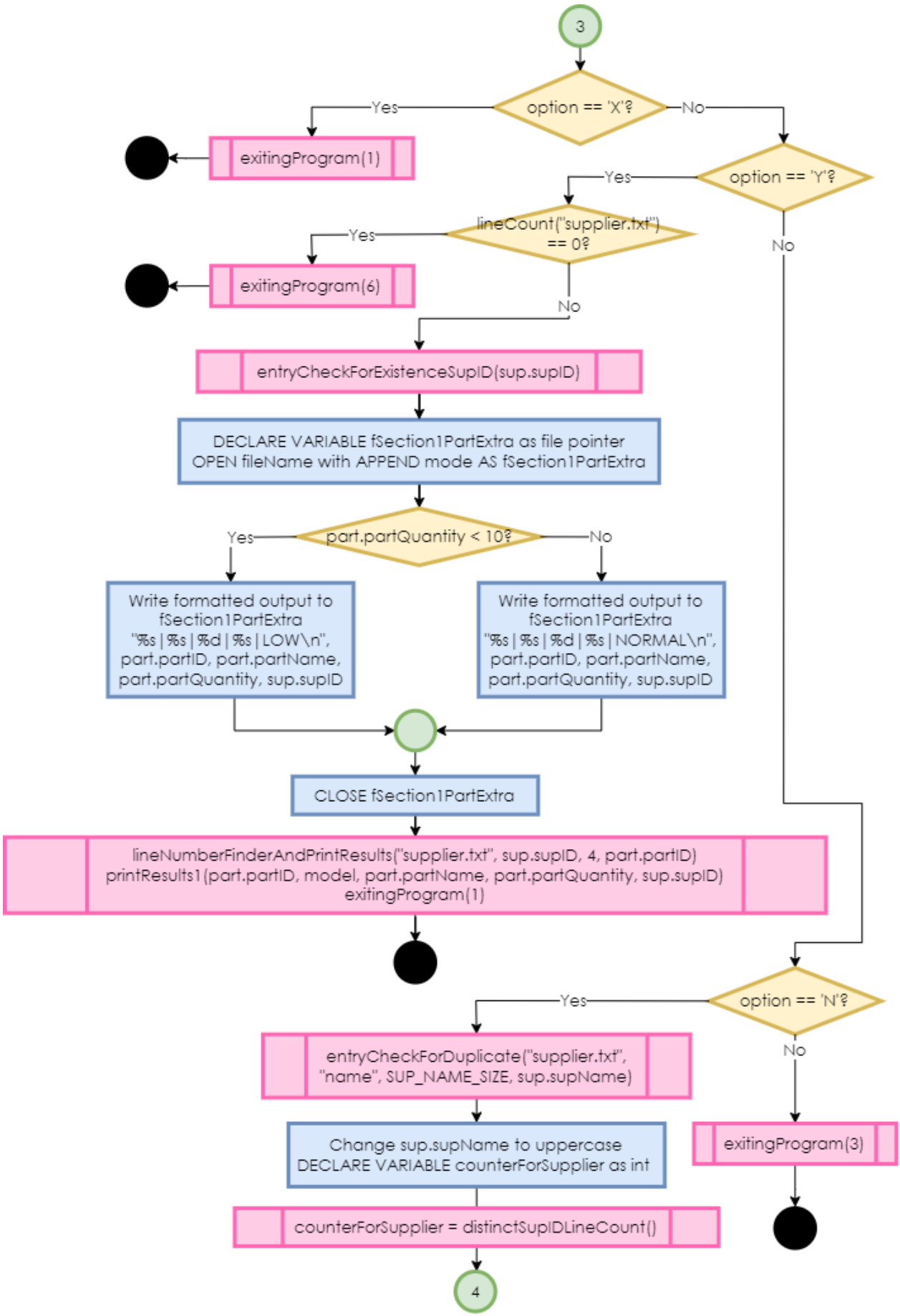
int distinctSupIDLineCount()

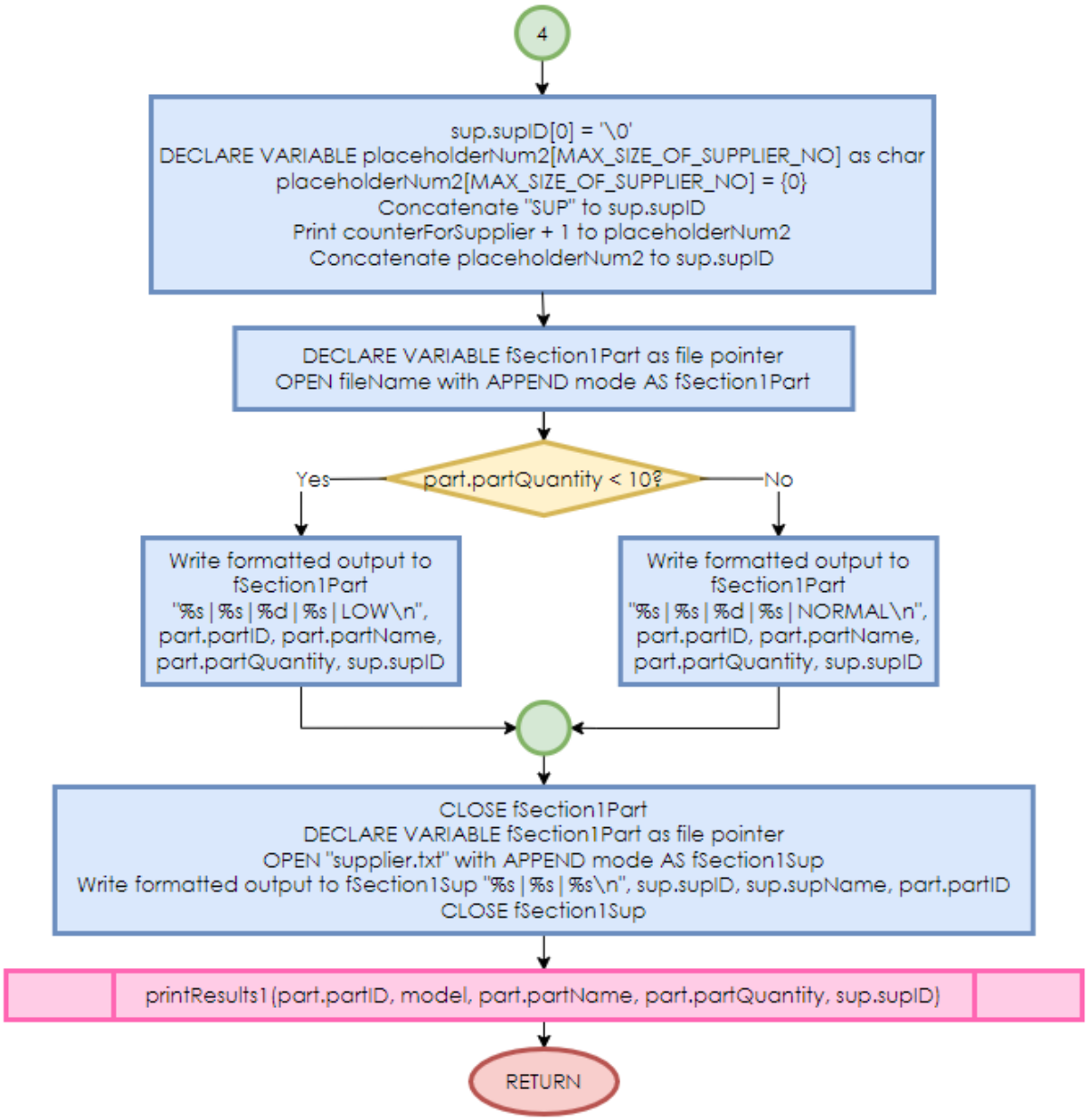


void partsInventoryCreation()



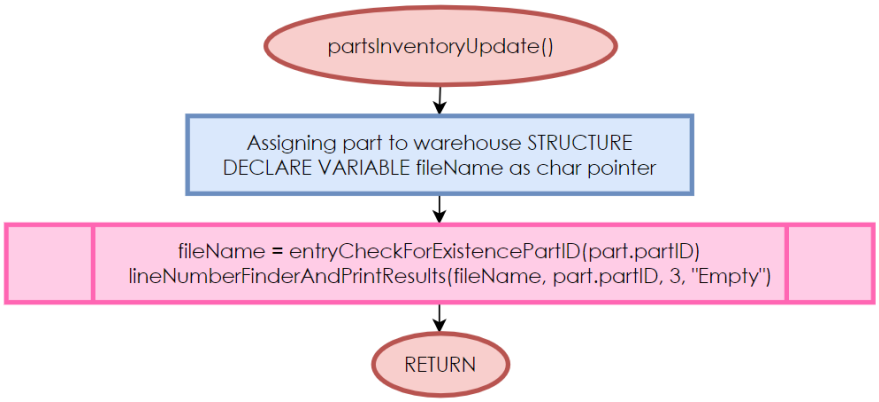

2
part.partQuantity = '\0'
Display "Please insert the quantity of part (", part.partName, ") that you would like to register"
Display "Enter negative numbers to quit:"
Read part.partQuantity
part.partQuantity < 0?
Yes
exitingProgram(1)
No
printingSeparator(69, '-')
Display newline
DECLARE VARIABLE counterForPart as int
counterForPart = 0
counterForPart = lineCount("WBZ.txt") + lineCount("WSL.txt") + lineCount("WAR.txt")
part.partID[0] = '\0'
DECLARE VARIABLE placeholderNum[MAX_SIZE_OF_PART_NO] as char
placeholderNum[MAX_SIZE_OF_PART_NO] = {0}
Concatenate part.warehouseChoice to part.partID
Print counterForPart + 1 to placeholderNum
Concatenate placeholderNum to part.partID
Assigning sup to supplier STRUCTURE
DECLARE VARIABLE option as char
Display "Do you have an existing Supplier ID? [Y (Yes) / N (No) / X (To quit)]"
Read option
Change option to uppercase
printingSeparator(69, '-')
3


3
option == 'X'?
Yes
No
exitingProgram(1)
option == 'Y'?
Yes
No
lineCount("supplier.txt") == 0?
Yes
No
exitingProgram(6)
entryCheckForExistenceSupID(sup.supID)
DECLARE VARIABLE fSection1PartExtra as file pointer
OPEN fileName with APPEND mode AS fSection1PartExtra
part.partQuantity < 10?
Yes
No
Write formatted output to fSection1PartExtra "%s | %s | %d | %s | LOW\n", part.partID, part.partName, part.partQuantity, sup.supID
Write formatted output to fSection1PartExtra "%s | %s | %d | %s | NORMAL\n", part.partID, part.partName, part.partQuantity, sup.supID
CLOSE fSection1PartExtra
lineNumberFinderAndPrintResults("supplier.txt", sup.supID, 4, part.partID)
printResults1(part.partID, model, part.partName, part.partQuantity, sup.supID)
exitingProgram(1)
option == 'N'?
Yes
No
entryCheckForDuplicate("supplier.txt", "name", SUP_NAME_SIZE, sup.supName)
exitingProgram(3)
Change sup.supName to uppercase
DECLARE VARIABLE counterForSupplier as int
counterForSupplier = distinctSupIDLineCount()
4

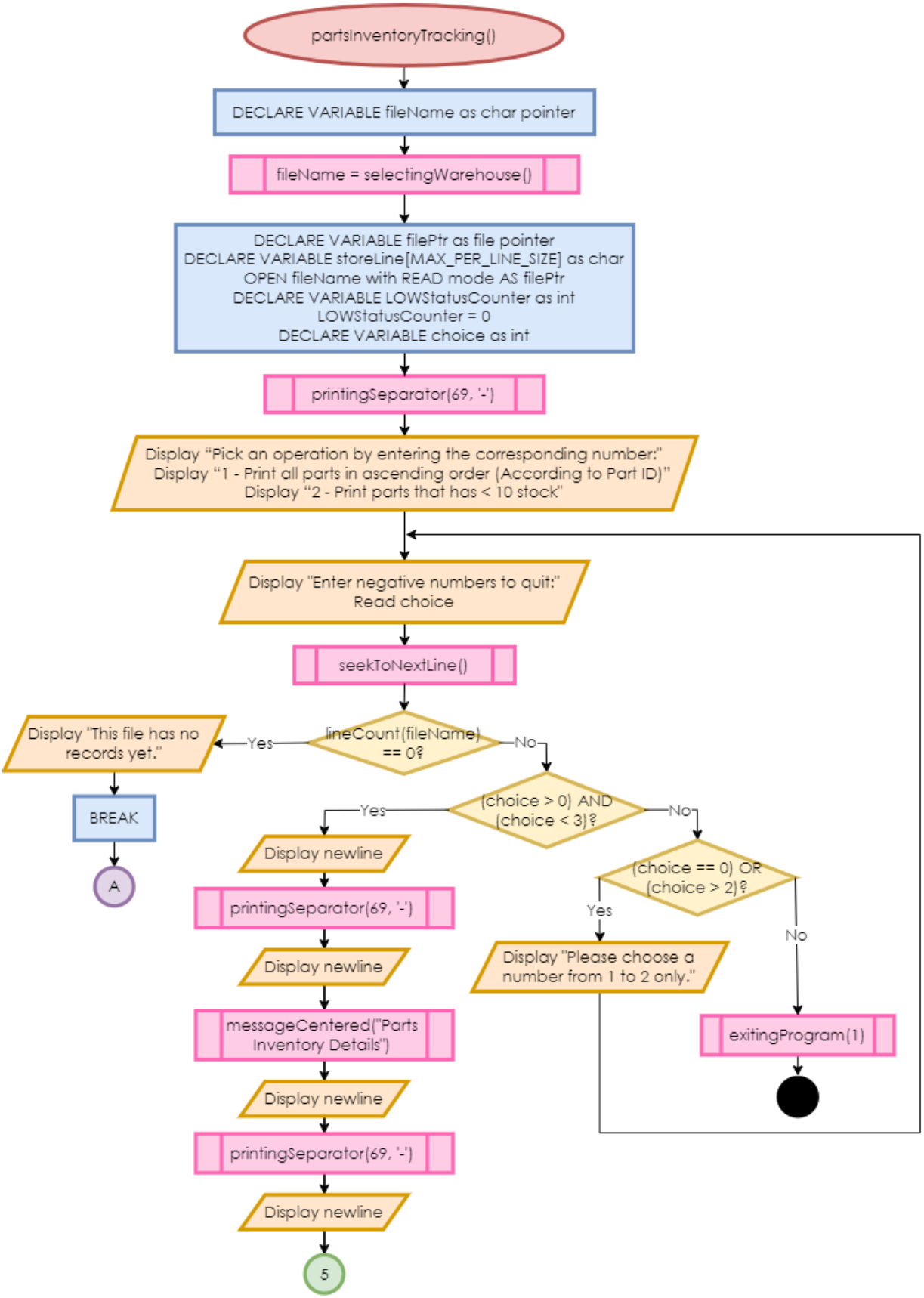


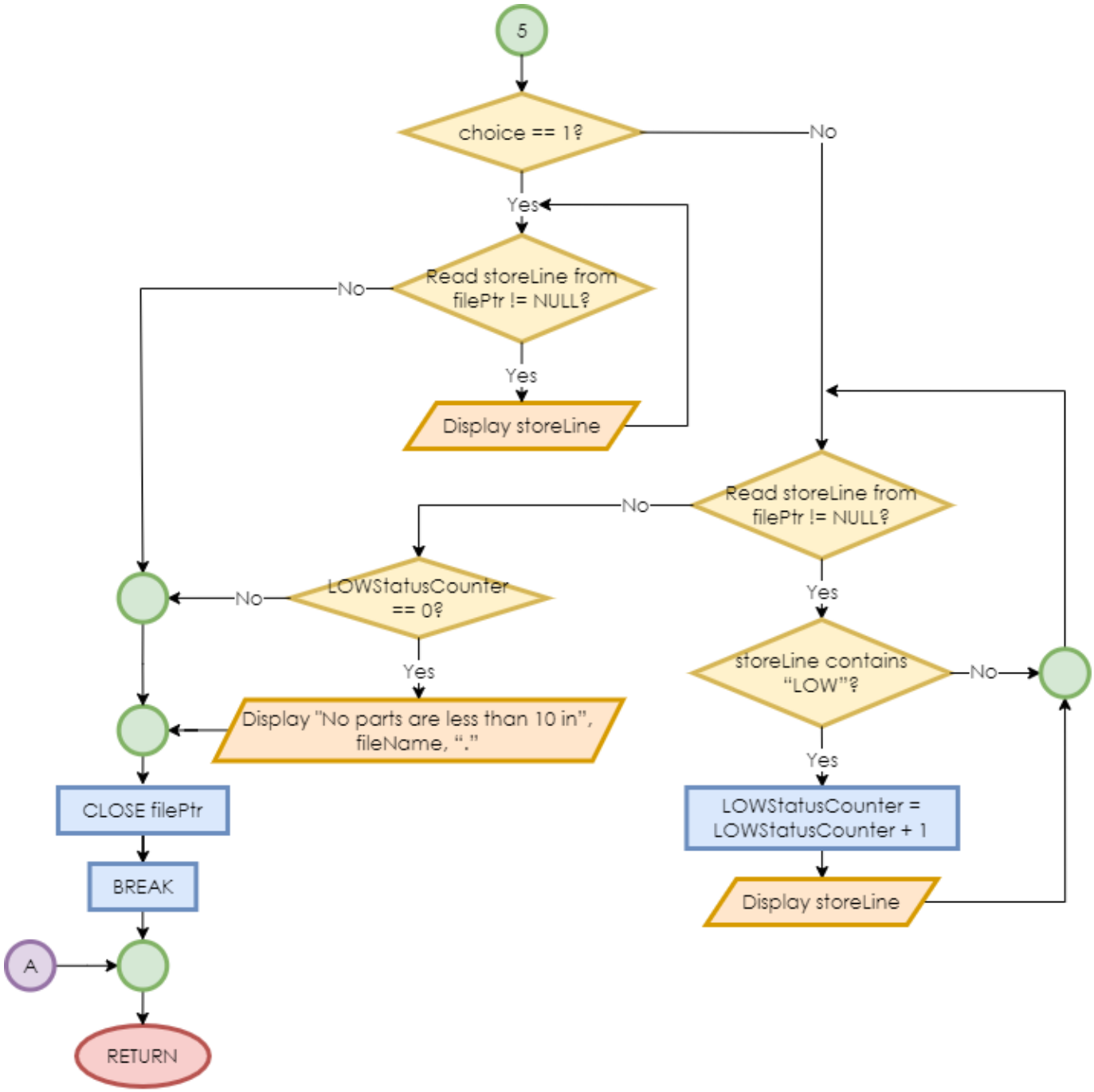
SECTION 2 FUNCTIONS

void partsInventoryUpdate()



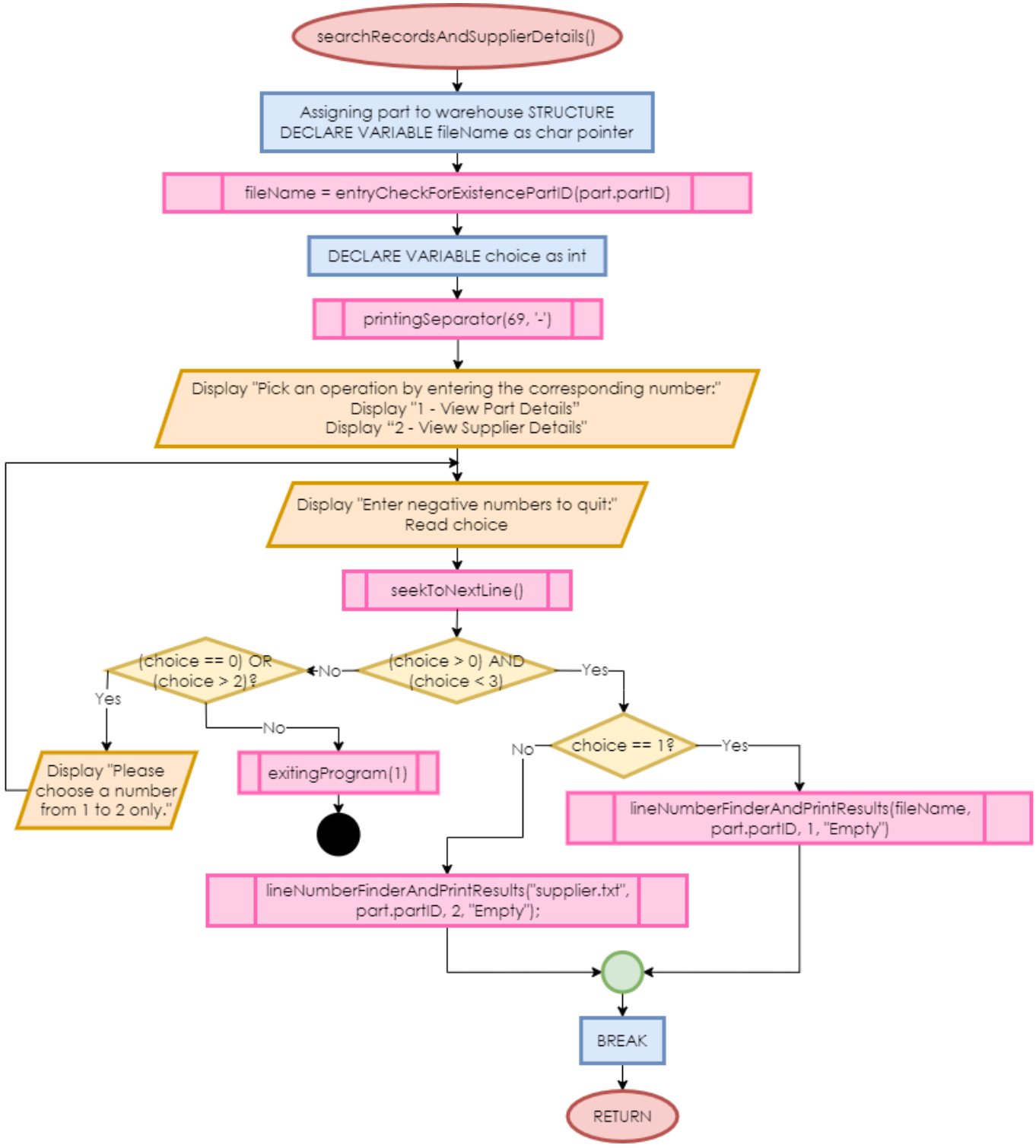
void partsInventoryTracking()



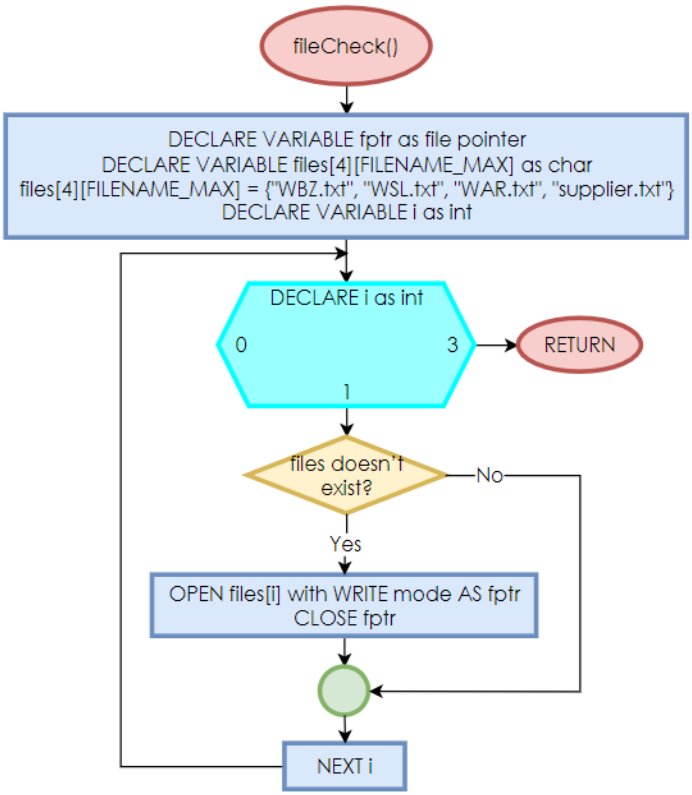

5
choice == 1?
No
Yes
Read storeLine from filePtr != NULL?
No
Yes
Display storeLine
Read storeLine from filePtr != NULL?
No
Yes
LOWStatusCounter == 0?
No
Yes
Display "No parts are less than 10 in", fileName, "."
storeLine contains "LOW"?
No
Yes
LOWStatusCounter = LOWStatusCounter + 1
Display storeLine
CLOSE filePtr
BREAK
A
RETURN

SECTION 4 FUNCTIONS

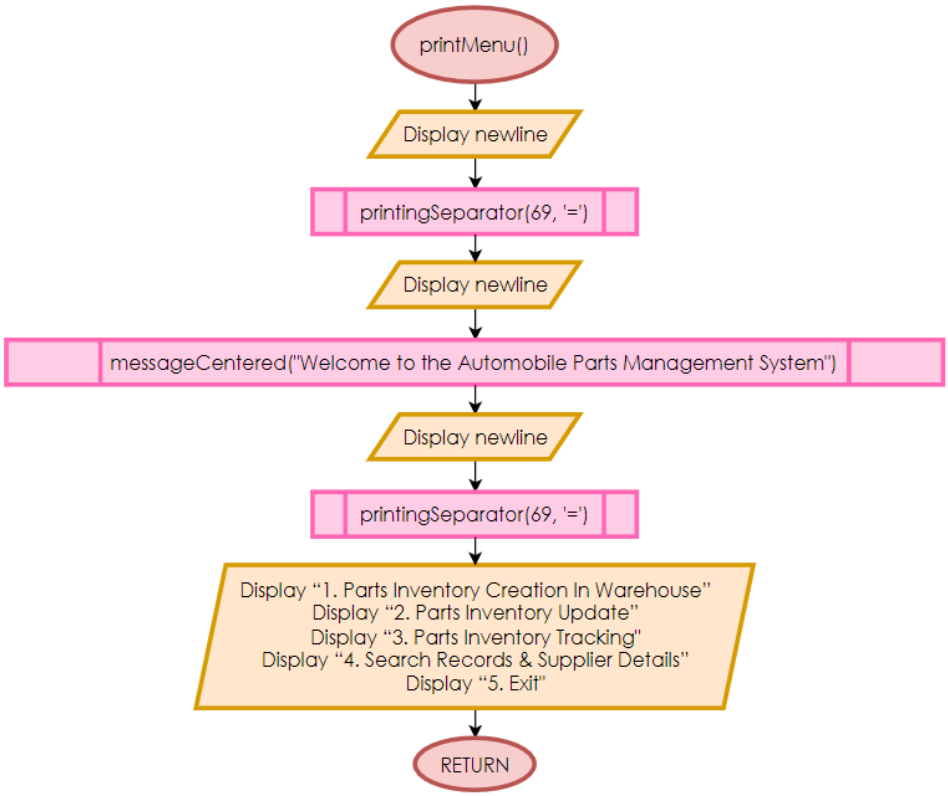
void searchRecordsAndSupplierDetails()



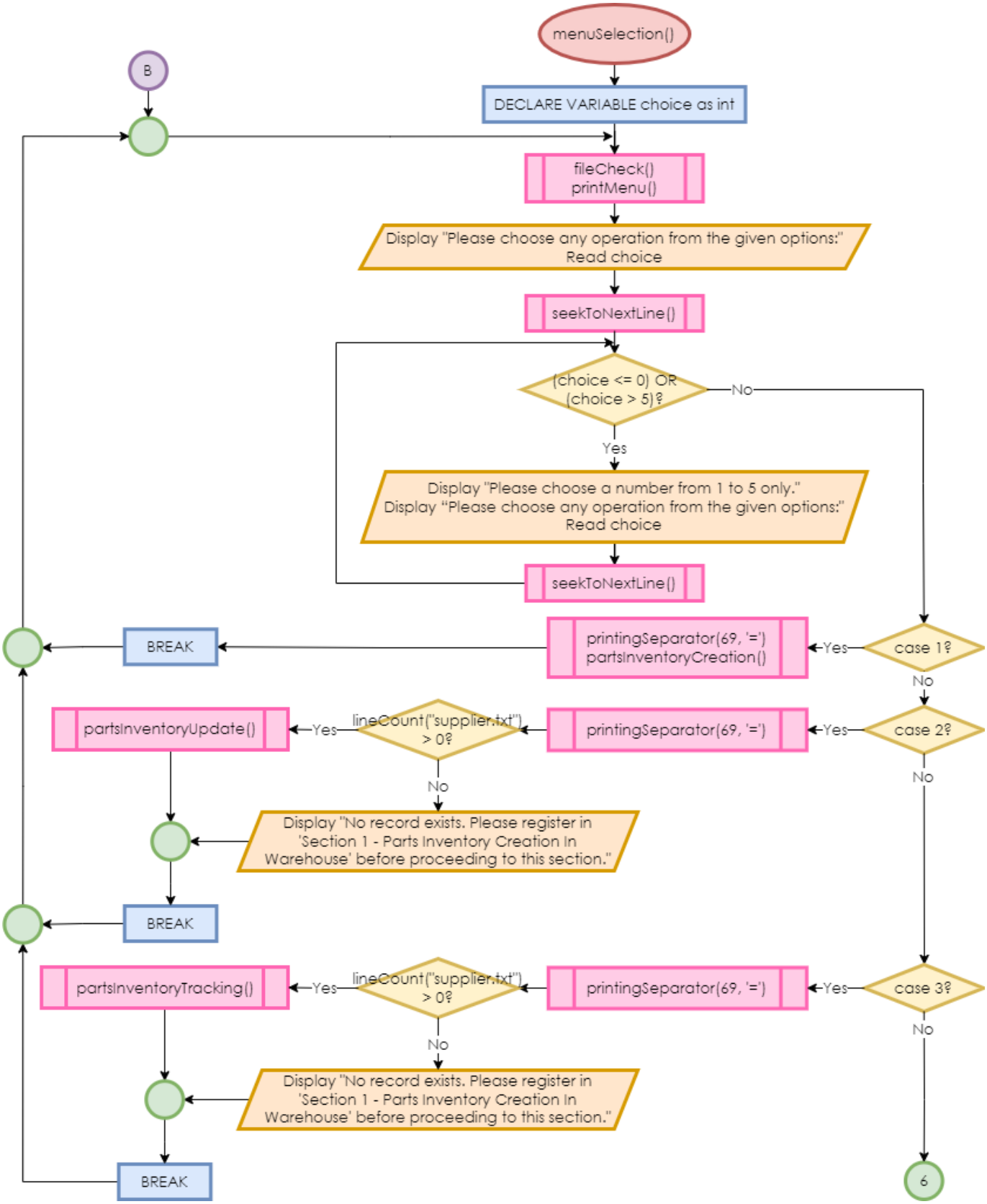
void fileCheck()

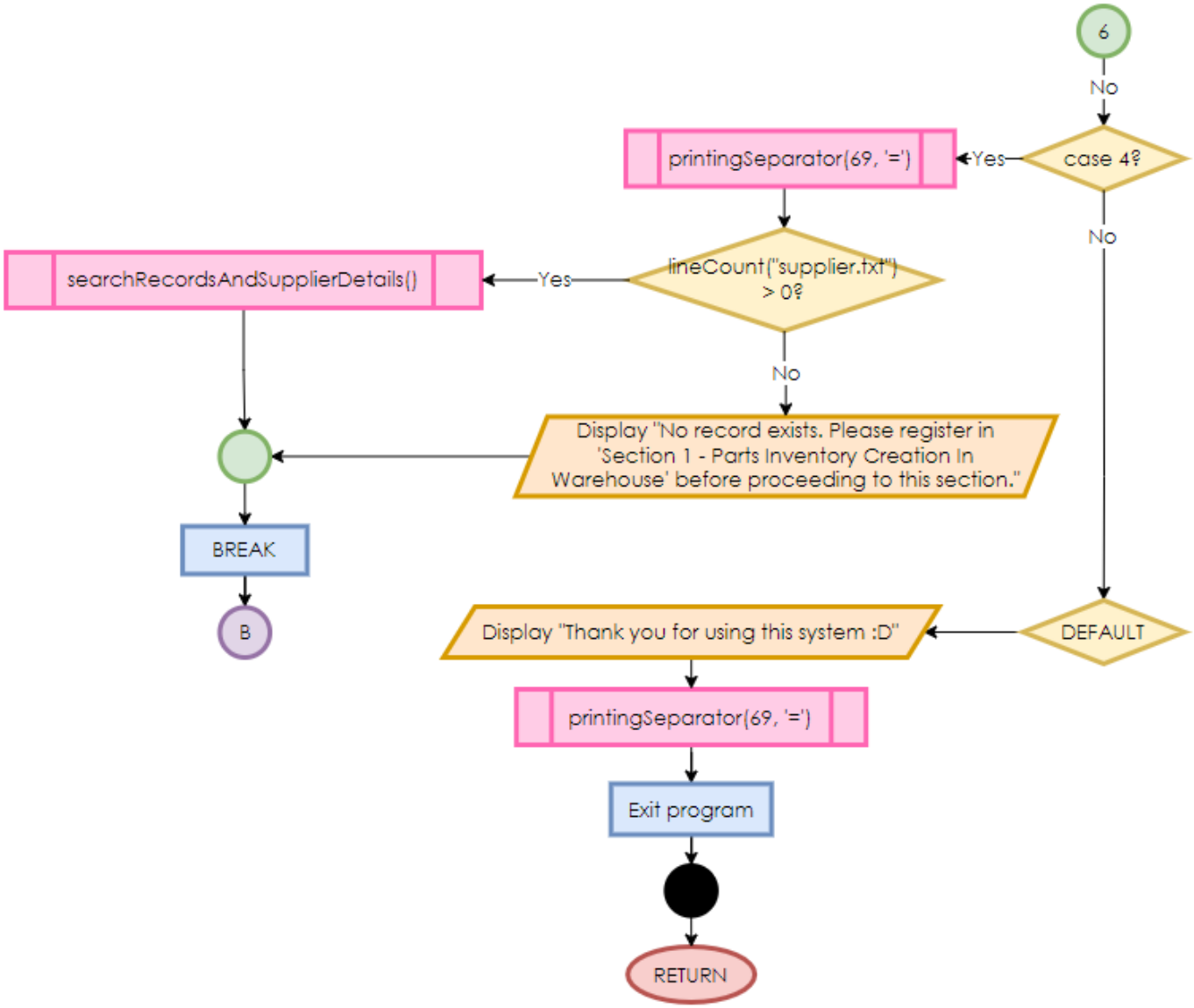


void printMenu()

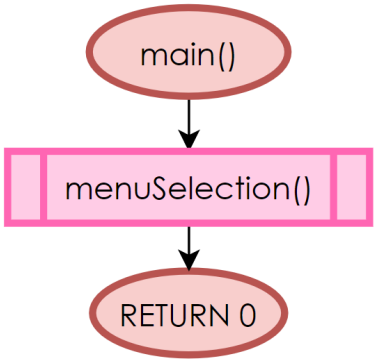


void menuSelection()

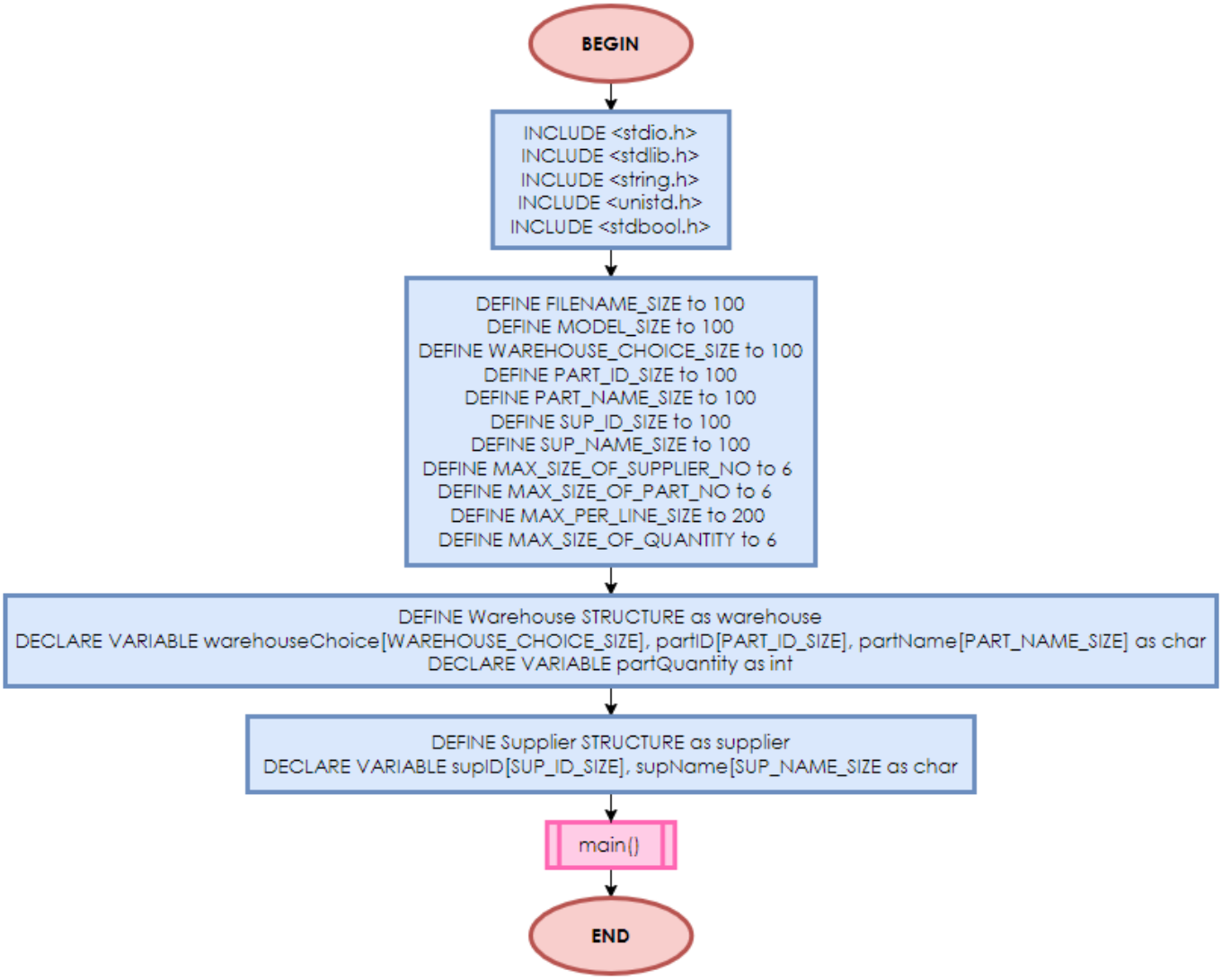



6
No
case 4?
Yes
printingSeparator(69, '=')
No
lineCount("supplier.txt") > 0?
Yes
searchRecordsAndSupplierDetails()
No
Display "No record exists. Please register in 'Section 1 - Parts Inventory Creation In Warehouse' before proceeding to this section."
BREAK
B
DEFAULT
Display "Thank you for using this system :D"
printingSeparator(69, '=')
Exit program
RETURN

int main()


main()
menuSelection()
RETURN 0

MAIN CODE


BEGIN
INCLUDE <stdio.h>
INCLUDE <stdlib.h>
INCLUDE <string.h>
INCLUDE <unistd.h>
INCLUDE <stdbool.h>
DEFINE FILENAME_SIZE to 100
DEFINE MODEL_SIZE to 100
DEFINE WAREHOUSE_CHOICE_SIZE to 100
DEFINE PART_ID_SIZE to 100
DEFINE PART_NAME_SIZE to 100
DEFINE SUP_ID_SIZE to 100
DEFINE SUP_NAME_SIZE to 100
DEFINE MAX_SIZE_OF_SUPPLIER_NO to 6
DEFINE MAX_SIZE_OF_PART_NO to 6
DEFINE MAX_PER_LINE_SIZE to 200
DEFINE MAX_SIZE_OF_QUANTITY to 6
DEFINE Warehouse STRUCTURE as warehouse
DECLARE VARIABLE warehouseChoice[WAREHOUSE_CHOICE_SIZE], partID[PART_ID_SIZE], partName[PART_NAME_SIZE] as char
DECLARE VARIABLE partQuantity as int
DEFINE Supplier STRUCTURE as supplier
DECLARE VARIABLE supID[SUP_ID_SIZE], supName[SUP_NAME_SIZE as char
main()
END

PROGRAM SOURCE CODE AND SAMPLE INPUT/OUTPUT

Note: “END:” indicates the input that ends the program.

==========

GENERAL FUNCTIONS

Functions that are used in more than one section.

==========

void printingSeparator(int length, char item)

```
38      // Print customized separator
39      void printingSeparator(int length, char item) {
40          for (int i = 0; i < length; i++)
41              printf("%c", item);
42      }
```

- To print a customized separator using a for loop.
- The item variable is a character that will be duplicated multiple times to form a separator.
- The length of the line (separator) or the number of characters printed is indicated by the length variable.

---

int lineCount(char fileUsed[FILENAME_SIZE])

```
44      // To count how many lines are in the file
45      int lineCount(char fileUsed[FILENAME_SIZE]) {
46          int lineCount = 0;
47          char chr;
48          FILE *fptr;
49          fptr = fopen(fileUsed, "r");
50
51          chr = getc(fptr);                   // Extract character from file and store in chr
52          while (chr != EOF) {
53              if (chr == '\n')                // Count whenever a new line is encountered
54                  lineCount = lineCount + 1;
55              chr = getc(fptr);               // Take next character from file
56          }
57
58          fclose(fptr);
59          return lineCount;
60      }
```

- To count how many lines are in a file.

- A file pointer is declared, then a file used as an argument will be opened in read mode.
- getc( ) will be used to extract the character from the file and store in the chr variable. The line counter (lineCount variable) will increase by 1 whenever a newline is encountered.
- Another getc( ) will be used to get the next character and this process repeats in a while loop until the end of the file is reached.
- The file will be closed and lineCount will be returned to indicate the number of lines in the file provided.

---

void seekToNextLine()

```
62    // Clears input buffer -> To prevent scanf() duplicate when >1 characters are inserted
63    void seekToNextLine() {
64        int c;
65        while((c = fgetc(stdin)) != EOF && c != '\n');
66    }
```

- To clear the input buffer.
- This code is a while loop that will read input as long as it is not the end of the file or it is not a new line.
- Helps to prevent scanf( ) reading input multiple times when more than 1 character is inserted.

---

void messageCentered(char *message)

```
68    // Make sure the text printed is in the center
69    void messageCentered(char *message) {
70        int length;
71        int position;
72
73        length = (69 - strlen(message)) / 2; // Calculate the number of spaces that need to be printed
74
75        for(position = 0; position < length; position++)
76            printf(" ");                     // Print that many spaces
77
78        printf("%s", message);               // Print the messages displayed through the functions
79    }
```

- To make sure a text will be printed in the centre of the output.
- The message variable stores the message the user wishes to print, which I mainly used for titles and table headers.
- In CLion, I found out that the maximum length of a row is 69 characters since there will always be abnormal outputs when you exceed 69 characters. Hence, the code will receive a string message as

an argument, then be subtracted by 69, and divided by 2 since the number of spaces only has to be printed on the left side (Before the message) only.

- Uses a for loop to print the spaces, and finally prints the message stored in the message variable.

---

void exitingProgram(int status)

```c
// Exiting program message
void exitingProgram(int status) {
    if (status == 1)
        printf("Thank you for using this system :D\n");
    if (status == 2)
        printf("Entry does not exist. Exiting program...\n");
    if (status == 3)
        printf("Invalid input. Exiting program...\n");
    if (status == 4)
        printf("Error opening files(s). Exiting program...\n");
    if (status == 5)
        printf("Entry already exists. Exiting program...\n");
    if (status == 6)
        printf("No records exist. Exiting system...\n");
    printingSeparator( length: 69,  item: '=');
    exit(0);
}
```

- To print an exiting program message according to different situations and exits program.
- The different situations differ according to the value of the status variable which is an integer.

```
Please insert your Part ID
Enter 'X' to quit:x
Thank you for using this system :D
=====================================================================
Process finished with exit code 0
```

Output: Sample Exit Indication

---

int entryExists(char *entry, FILE *file)

```c
// Check if the entry already exists (partFinder == 1) in the file chosen
int entryExists(char *entry, FILE *file) {
    int partFinder = 0;
    char lineInFile[MAX_PER_LINE_SIZE];

    while (fgets(lineInFile, MAX_PER_LINE_SIZE, file) != NULL) {    // Check by line
        if ((strstr(lineInFile, entry)) != NULL)                    // Increases partFinder when the entry exists
            partFinder++;
        if (partFinder == 1)       // Part exists
            return partFinder;
    }
    return partFinder;             // Part does not exist
}
```

- To check if the entry already exists by returning 1 via the variable partFinder.
- If the entry does not exist, it returns 0.
- The while loop will check line by line of the file until it is NULL.
- Line 105 checks if the entry variable exists in the line of a file using strstr( ) as it finds the first occurrence of the substring (entry) in the lineInFile variable.
- If the entry variable is found within a line in the file, it increases by 1 and will be returned.

---

char *entryCheckForExistencePartID(char *entryPartID)

```
113  // Check if Part ID exists and return its respective filename, if no, exits program
114  char *entryCheckForExistencePartID(char *entryPartID) {
115      int flag = 0;
116      while (1) {
117          printf("Please insert your Part ID");
118          printf("\nEnter 'X' to quit:");
119          scanf("%[^\n]s", entryPartID);                          // Getting input until a newline is scanned
120          strupr(entryPartID);
121          seekToNextLine();
122
123          if (strcmpi(entryPartID, "X") == 0) {                   // If input is "X", exit program
124              exitingProgram( status: 1);
125          } else if (strlen(entryPartID) > PART_ID_SIZE) {        // Validating input length
126              printf("\nPlease input your Part ID within %d characters.\n", PART_ID_SIZE);
127              continue;
128          } else if (lineCount( fileUsed: "supplier.txt") > 0) {
129              // Extract warehouse info from partID to assign the respective warehouse .txt file
130              char warehouse[PART_ID_SIZE];
131              snprintf(warehouse, PART_ID_SIZE, "%.3s", entryPartID); // snprintf(target, sizeof target, "%.10s", source);
132
133              // Assigning the respective warehouse .txt file
134              char fileName[FILENAME_SIZE];
135              strcat(fileName, warehouse);
136              strcat(fileName, ".txt");
137
138              FILE *fptr;
139              fptr = fopen(fileName, "r");
140
```

```
141            /*
142             Check file content if user input item already exists, if no (flag = 0), exits program
143             flag = 1 (Exist in file)
144             flag = 0 (Doesn't exist in file)
145            */
146            flag = entryExists(entryPartID, fptr);
147            fclose(fptr);
148            if (flag == 1) {
149                if (strcmpi(fileName,"WBZ.txt") == 0){
150                    return "WBZ.txt";
151                }
152                if (strcmpi(fileName,"WSL.txt") == 0){
153                    return "WSL.txt";
154                }
155                if (strcmpi(fileName,"WAR.txt") == 0) {
156                    return "WAR.txt";
157                }
158            } else {
159                exitingProgram( status: 2); // Exit the program -> An entry doesn't exist
160            }
161        }
162    }
163 }
```

- To check if a Part ID exists or not and return its respective warehouse file name that it belongs to. It exits the program if the Part ID does not exist in any of the files. (Does not prompt for warehouse information as the system checks which warehouse the Part ID belongs to)
- entryPartID variable is used to store the partID inputted by the user.
- Line 117 - 121: Prints a menu (To prompt for Part ID), reads the input of the user and store it in a variable called entryPartID. It also converts the input into uppercase by strupr( ) so that input validations will be easier.
- Line 125 - 127: Asks user to reenter input when the input exceeds the length allocated for it.

```
Please insert your Part ID
Enter 'X' to quit:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Please input your Part ID within 100 characters.
Please insert your Part ID
Enter 'X' to quit:
```

Output: Length Validation

- Line 128 - 161 will only run when supplier.txt is not empty.
- Line 130 & 131: Extract warehouse info (First 3 letters) from partID to assign the respective warehouse .txt file using snprintf( ) which writes formatted input to a buffer. This is because the first three letters of the Part ID indicates the warehouse it belongs to.

- Line 134 - 136: Assigning the respective warehouse .txt file to the fileName variable by using strcat( ) to concatenate ".txt" to the warehouse name.
- Line 138 - 161: Check file content to see if the user input item already exists, if no (flag = 0), it exits the program because the entry does not appear in any of the 3 warehouse files. If yes (flag = 1), it returns the file name that it belongs to by using strcmpi( ) to compare the filename with the available file names.
- When strcmpi( )'s two input matches completely, it will return 0, hence I used == 0 as the condition.

```
Please insert your Part ID
Enter 'X' to quit:test
Entry does not exist. Exiting program...
========================================================================
Process finished with exit code 0
```

Output: Invalid Part ID Validation

- END: X or x

---

void entryCheckForDuplicate(char fileUsed[FILENAME_SIZE], char item[10], int maxSizeOfItem, char *entry)

```
165  // Check if an entry already exists and exits the program if so
166  void entryCheckForDuplicate(char fileUsed[FILENAME_SIZE], char item[10], int maxSizeOfItem, char *entry) {
167      int flag = 1;
168      FILE *fptr;
169      fptr = fopen(fileUsed, "r");
170
171      while (1) {
172          printf("Please insert the %s that you want to register", item);
173          printf("\nEnter 'X' to quit:");
174          scanf("%[^\n]s", entry);                        // Getting input until a newline is scanned
175          strupr(entry);
176          seekToNextLine();
177
178          if (strcmpi(entry, "X") == 0) {                 // If input is "X", exit program
179              exitingProgram( status: 1);
180          } else if (strlen(entry) > maxSizeOfItem) {      // Validating input length
181              printf("\nPlease input a %s within %d characters.\n", item, maxSizeOfItem);
182              continue;
183          } else if (lineCount(fileUsed) > 0) {
184              /*
185               Check file content if user input item already exists, if yes (flag = 1), exits program
186               flag = 1 (Exist in file)
187               flag = 0 (Doesn't exist in file)
188              */
189              flag = entryExists(entry, fptr);
190              fclose(fptr);
191              if (flag == 0)
```

```
192                 break;
193             else
194                 exitingProgram( status: 5); // Exiting program -> Entry already exists
195         } else
196             break;
197     }
198 }
```

- To check if an entry inputted already exists and exits the program if so.
- entry variable is used to store the input by the user.
- Line 167 - 169: Creates a file pointer and open the file provided as an argument for reading.
- Line 172 - 178: Prints a menu (To prompt for input), reads the input of the user and store it in a variable called entry. It also converts the input into uppercase by strupr( ) so that input validations will be easier.
- Line 180 - 182: Asks user to reenter input when the input exceeds the length allocated for it.

```
Please insert the part that you want to register
Enter 'X' to quit:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Please input a part within 100 characters.
Please insert the part that you want to register
Enter 'X' to quit:
```

Output: Length Validation

- Line 183 - 194 will only run when the file provided as an argument is not empty.
- Line 189 - 194: Check file content to see if the user input item already exists, if no (flag = 0), it breaks the while loop and will go back to the main function that it is called from. If yes (flag = 1), it exits the program since the program will not accept duplicate entries for certain sections in the files.

```
Please insert the part that you want to register
Enter 'X' to quit:mirror
Entry already exists. Exiting program...
======================================================================
Process finished with exit code 0
```

Output: Existing Entry Validation

- If the file provided as an argument is empty, it will just go back to the main function that it is called from straightaway since there will be no point checking the files.
- END: X or x

void replaceLine(char fileUsed[FILENAME_SIZE], int lineNum, char replaceText[MAX_PER_LINE_SIZE])

```
200     // Replace the line
201     // Write all the lines of the original file to a temp file EXCEPT the line we want to replace (Write a replacement line in its place)
202     // Delete the original file
203     // Rename the temp file to the original file's name
204     void replaceLine(char fileUsed[FILENAME_SIZE], int lineNum, char replaceText[MAX_PER_LINE_SIZE]) {
205         FILE *filePtr, *tempPtr;
206         char buffer[MAX_PER_LINE_SIZE];                 // Buffer will store each line from the original file
207
208         // Open file for reading and writing
209         filePtr = fopen(fileUsed, "r");
210         tempPtr = fopen("tempFile.txt", "w");           // Making the temporary file
211
212         if (filePtr == NULL || tempPtr == NULL)         // Check if either file failed to open, if either did exit with error status
213             exitingProgram( status: 4);                 // Exit program -> Error opening files
214
215         bool keepReading = true;
216         int currentLine = 1;
217         do {
218             fgets(buffer, MAX_PER_LINE_SIZE, filePtr);  // Read the next line of the file into the buffer
219             if (feof(filePtr))
220                 keepReading = false;                    // Stop reading if EOF reached
221             else if (currentLine == lineNum)
222                 fputs(replaceText, tempPtr);            // Write the replacement text to this line of the temp file if the line we've reached is the one we wish to replace
223             else
224                 fputs(buffer, tempPtr);                 // Otherwise, write this line to the temp file
225             currentLine++;                              // Increment to read the next line
226         } while (keepReading);
227
228         fclose(filePtr);
229         fclose(tempPtr);
230
231         remove(fileUsed);                               // Delete the original file
232         rename("tempFile.txt", fileUsed);               // Rename temp file to the original file's name
233     }
```

- Replace a line in a file by writing all the lines of the original file to a temporary file except the line I want to replace (Write the replacement line in its place), deletes the original file, then renames the temporary file to the original file's name.
- Buffer variable is used to store each line from the original file.
- Line 209 & 210: Open the file provided as argument for reading and a temporary file (“tempFile.txt”) for writing.
- Line 212 - 213: Check if either file failed to open, if either did, it will exit the program.
- Line 215 - 229: Reads the lines into the buffer while keepReading variable is true. If the current line matches the line number given as an argument, it will write the replacement text to this line of the temporary file since the line is the one we wish to replace. Otherwise, it will write the original lines that already exist in the file originally. currentLine will keep track of the current line number. If the file pointer reaches the end of file, keepReading will be false which breaks the while loop, and it will proceed to the next line of codes.
- Line 228 & 229 - Closes the files to avoid further modification.
- Line 231 & 232 - Deletes the original file and rename the temporary file to the original’s file name.

void updateStockQuantity(char partID[PART_ID_SIZE], char partName[PART_NAME_SIZE], char quantity[MAX_SIZE_OF_QUANTITY], char supID[SUP_ID_SIZE], char fileUsed[FILENAME_SIZE], int lineNum)

```
// Update stock quantity and generate the line to replace original row
void updateStockQuantity(char partID[PART_ID_SIZE], char partName[PART_NAME_SIZE], char quantity[MAX_SIZE_OF_QUANTITY],
                         char supID[SUP_ID_SIZE], char fileUsed[FILENAME_SIZE], int lineNum) {
    int currentQuantity, result, inputNum;
    currentQuantity = atoi(quantity);

    // Select from menu
    while (1) {
        int choice;
        printf("\nPick an operation by entering the corresponding number:");
        printf("\n\t1 - Increase stock quantity\n\t2 - Decrease stock quantity");
        printf("\nEnter negative numbers to quit:");
        scanf("%d", &choice);
        seekToNextLine();

        if (choice > 0 && choice < 3) {
            if (choice == 1) {
                printf("Please insert the stock amount to be added:");
                scanf("%d", &inputNum);
                seekToNextLine();
                result = currentQuantity + inputNum;  // Add inventory
            }
            if (choice == 2) {
                printf("Please insert the stock amount to be subtracted: ");
                scanf("%d", &inputNum);
                seekToNextLine();
                if (inputNum > currentQuantity) {
                    printf("\nCurrent stock is insufficient, unable to grant parts for assembly. Please try again.\n\n");
                    continue;
                }
                result = currentQuantity - inputNum;  // Subtract inventory
            }
            printf("\nDetails updated. Returning to main menu...\n");
            break;
        } else if (choice == 0 || choice > 2) {       // Validating input
            printf("Please choose a number from 1 to 2 only.\n");
        } else {
            exitingProgram( status: 1);                  // If input is a negative number, exit program
        }
    }

    // Store the line to be replaced
    char replaceText[MAX_PER_LINE_SIZE];
    if (result < 10)
        snprintf(replaceText, MAX_PER_LINE_SIZE,"%s|%s|%d|%s|LOW\n",partID, partName, result, supID);
    else
        snprintf(replaceText, MAX_PER_LINE_SIZE,"%s|%s|%d|%s|NORMAL\n",partID, partName, result, supID);

    replaceLine(fileUsed, lineNum, replaceText);
}
```

- To update the stock quantity and generate the line number that will replace the original row.
- atoi( ) is used to convert char to integer.

- Line 243 - 248: Prints a menu (To increase or decrease stock quantity), reads the input of the user and store it in a variable called choice.
- Line 251 - 256: Prompts for the user to insert the stock amount to be added and does the calculation that adds the user input with the original amount. The sum is placed in a variable called result.
- Line 257 - 266: Prompts for the user to insert the stock amount to be subtracted and does the calculation that subtracts the user input from the original amount. However, if the current amount of stock is lower than the user input, the system will ask the user to reenter input without doing the calculations. Otherwise, the subtracted value will be placed in a variable called result.

```
Please insert the stock amount to be subtracted:1000

Current stock is insufficient, unable to grant parts for assembly. Please try again.


Pick an operation by entering the corresponding number:
        1 - Increase stock quantity
        2 - Decrease stock quantity
Enter negative numbers to quit:
```

Output: Insufficient Stock Validation

- Line 269 & 270: If the user input for the choice variable is 0 or more than 2 (Values that were not shown in the menu), it will ask the user to reenter input.

```
Pick an operation by entering the corresponding number:
        1 - Increase stock quantity
        2 - Decrease stock quantity
Enter negative numbers to quit:5
Please choose a number from 1 to 2 only.

Pick an operation by entering the corresponding number:
        1 - Increase stock quantity
        2 - Decrease stock quantity
Enter negative numbers to quit:m
Please choose a number from 1 to 2 only.

Pick an operation by entering the corresponding number:
        1 - Increase stock quantity
        2 - Decrease stock quantity
Enter negative numbers to quit:
```

Output: Incorrect Input Validation

- EXIT: Any negative numbers

- Line 277 - 283: replaceText variable will store the line to be replaced. If the result variable is less than 10, the status part of the text to be replaced will be “LOW”. Otherwise, it will be “NORMAL”. The line will be replaced in the *replaceLine function*.

---

void lineNumberFinderAndPrintResults(char fileUsed[FILENAME_SIZE], char ID[PART_ID_SIZE], int choice, char extraPartForSection1[PART_ID_SIZE])

```
285     // Check which line number the entry is in and print/write the details using tokens
286     void lineNumberFinderAndPrintResults(char fileUsed[FILENAME_SIZE], char ID[PART_ID_SIZE], int choice, char extraPartForSection1[PART_ID_SIZE]) {
287         FILE *filePtr;
288         int lineNum = 0;
289         int findPart = 0;
290         char lineInFile[MAX_PER_LINE_SIZE];
291 
292         filePtr = fopen(fileUsed, "r");
293 
294         while (fgets(lineInFile, MAX_PER_LINE_SIZE, filePtr) != NULL) {     // Check by line
295             if ((strstr(lineInFile, ID)) != NULL) {                         // Increases findPart when the ID exists
296                 findPart++;
297             }
298             lineNum++;
299             if (findPart == 1) {                                            // Part exists
300                 break;
301             }
302         }
303         fclose(filePtr);
```

- Check which line number the entry is in and print the details using tokens.
- Line 287 - 303: A file pointer is declared and a file used as an argument will be opened for reading purposes. A while loop will check line by line to see if the ID which is the argument passed exists in the file. If yes, findPart will increase and breaks the while loop. The ID will definitely exist in the file as the existence of the ID will be validated in previous functions. The file is then closed to avoid further modifications.

```
305         // To convert string in file to token
306         int x = 0;
307         char *array[5];
308         lineInFile[strcspn(lineInFile, "\n")] = 0; // Set the length of the number of characters per line before \n in []
309                                                   // That index will be set to 0 -> Marks the end of a line
310 
311         // Extract the first token
312         char *token = strtok(lineInFile, "|");     // strtok() -> Breaks string into a series of tokens using a | delimiter
313         token[strcspn(token, "\n")] = 0;           // strcspn() -> Calculates the length of the number of characters before the
314                                                   //              1st occurrence of character present in the string
315                                                   // That index will be set to 0 -> Marks the end of a token
316 
317         // Loop through the string to extract all other tokens
318         while (token != NULL) {
319             array[x] = token;
320             x++;
321             token = strtok(NULL, "|");            // Stop condition to break the loop
322         }
```

- Line 305 - 320: To convert strings in a file to token using strtok( ) and strcspn( ). strtok( ) will breaks the string (lineInFile) into a series of tokens using a “|” delimiter. strcspn( ) will calculate

the length of the number of characters before the first occurrence of character present in the string, which in this case is newline. Then, it will loop through the string to extract all other tokens.

```
323         printf("\n");
324         printingSeparator( length: 69,  item: '-');
325         printf("\n");
326         if ((choice == 1) || (choice == 3)) {
327             // Display Part Details for Section 2 and 4
328             messageCentered( message: "Part Details");
329             printf("\n");
330             printingSeparator( length: 69,  item: '-');
331             printf("\n\tPart ID           = %s", array[0]);
332             printf("\n\tPart Name         = %s", array[1]);
333             printf("\n\tPart Quantity     = %s", array[2]);
334             printf("\n\tPart Status       = %s", array[4]);
335             printf("\n\tSupplier ID       = %s\n", array[3]);
336             printingSeparator( length: 69,  item: '-');
337             if (choice == 3) {
338                 updateStockQuantity( partID: array[0],  partName: array[1],  quantity: array[2],  supID: array[3], fileUsed, lineNum);
339             }
340         }
```

- Line 328 - 336: Prints the part details if the argument for choice variable is 1 or 3.
- Line 337 - 339: Updates the stock for a part in the *updateStockQuantity function* if the argument for choice variable is 3.

```
341         if (choice == 2) {
342             // Display Supplier Details for Section 4
343             messageCentered( message: "Supplier Details");
344             printf("\n");
345             printingSeparator( length: 69,  item: '-');
346             printf("\n\tPart ID           = %s", array[2]);
347             printf("\n\tSupplier ID       = %s", array[0]);
348             printf("\n\tSupplier Name     = %s\n", array[1]);
349             printingSeparator( length: 69,  item: '-');
350         }
351         if (choice == 4) {
352             // Write Supplier Details to supplier.txt for Section 1
353             filePtr = fopen(fileUsed, "a");
354             fprintf(filePtr, "%s|%s|%s\n", array[0], array[1], extraPartForSection1);
355             fclose(filePtr);
356         }
357     }
```

- Line 341 - 350: Prints the supplier details if the argument for choice variable is 2.
- Line 351 - 356: Write Supplier Details to supplier.txt for Section 1 if the argument for choice variable is 4.

char *selectingWarehouse()

```
359     // Assigns the respective warehouse .txt file according to model chosen
360     char *selectingWarehouse() {
361         int choice;
362         printf("Pick a model by entering the corresponding number:");
363         printf("\n\t1 - Blaze (BZ)\n\t2 - Silk (SL)\n\t3 - Armer (AR)");
364         while (1) {
365             printf("\nEnter negative numbers to quit:");
366             scanf("%d", &choice);
367             seekToNextLine();
368             if (choice > 0 && choice < 4) {
369                 if (choice == 1)
370                     return "WBZ.txt";    // Blaze warehouse
371                 if (choice == 2)
372                     return "WSL.txt";    // Silk warehouse
373                 return "WAR.txt";        // Armer warehouse
374             } else if (choice == 0 || choice > 3) {
375                 printf("Please choose a number from 1 to 3 only.\n");  // Validating input
376                 continue;
377             } else {
378                 exitingProgram( status: 1);                            // If input is a negative number, exit program
379             }
380         }
381     }
```

- Assigns the respective warehouse .txt file according to the model chosen.
- Line 243 - 248: Prints a menu (To choose the model) and reads the input of the user and store it in a variable called choice.
- Line 368 - 373: Assigns the respective warehouse .txt file according to the choice selection.
- Line 374 - 376: If the user input for the choice variable is 0 or more than 3 (Values that were not shown in the menu), it will ask the user to reenter input.

```
Pick a model by entering the corresponding number:
        1 - Blaze (BZ)
        2 - Silk (SL)
        3 - Armer (AR)
Enter negative numbers to quit:6
Please choose a number from 1 to 3 only.

Enter negative numbers to quit:m
Please choose a number from 1 to 3 only.

Enter negative numbers to quit:
```

Output: Incorrect Input Validation

- EXIT: Any negative numbers

==========

SECTION 1 FUNCTIONS

Functions that are only used in Section 1 - Parts Inventory Creation.

==========

void printResults1(char *partID, char* model, char *partName, int partQuantity, char* supID)

```c
// Print Section 1 results
void printResults1(char *partID, char* model, char *partName, int partQuantity, char* supID) {
    printf("\n");
    printingSeparator( length: 69,  item: '-');
    printf("\n");
    messageCentered( message: "Inventory Creation Details");
    printf("\n");
    printingSeparator( length: 69,  item: '-');
    printf("\nYou have successfully registered %s.", partID);
    printf("\n\tModel Chosen      = %s", model);
    printf("\n\tPart ID           = %s", partID);
    printf("\n\tPart Name         = %s", partName);
    printf("\n\tPart Quantity     = %d", partQuantity);
    printf("\n\tSupplier ID       = %s\n", supID);
    printingSeparator( length: 69,  item: '-');
    printf("\nDetails are successfully printed.\n");
}
```

Prints the results for Section 1 which includes the model chosen, Part ID, part name, part quantity and Supplier ID.

---

void entryCheckForExistenceSupID(char *entrySupID)

```c
// Checks if Supplier ID exist, if not, exits program
void entryCheckForExistenceSupID(char *entrySupID) {
    int flag = 0;
    FILE *fptr;
    fptr = fopen("supplier.txt", "r");
    while (1) {
        printf("Please insert your Supplier ID");
        printf("\nEnter 'X' to quit:");
        scanf("%[^\n]s", entrySupID);                   // Getting input until a newline is scanned
        strupr(entrySupID);
        seekToNextLine();

        if (strcmpi(entrySupID, "X") == 0) {            // If input is "X", exit program
            exitingProgram( status: 1);
        } else if (strlen(entrySupID) > SUP_ID_SIZE) {  // Validating input length
            printf("\nPlease input a Supplier ID within %d characters.\n", SUP_ID_SIZE);
            continue;
        } else if (lineCount( fileUsed: "supplier.txt") > 0) {
            /*
             Check file content if user input item already exists, if no (flag = 0), exits program
             flag = 1 (Exist in file)
             flag = 0 (Doesn't exist in file)
            */
            flag = entryExists(entrySupID, fptr);
            fclose(fptr);
            if (flag == 1)
                break;
```

```
430                     else
431                         exitingProgram( status: 2); // Exit the program -> An entry doesn't exist
432                 } else
433                     exitingProgram( status: 6);     // Exit the program -> Records doesn't exist
434             }
435         }
```

- To check if a Supplier ID exists or not. It exits the program if the Supplier ID does not exist in supplier.txt.
- entrySupID variable is used to store the Supplier ID inputted by the user.
- Line 406 - 407: A file pointer is declared and “supplier.txt” will be opened for reading purposes.
- Line 409 - 413: Prints a menu (To prompt for Supplier ID), reads the input of the user and store it in a variable called entrySupID. It also converts the input into uppercase by strupr( ) so that input validations will be easier.
- Line 417 - 419: Asks user to reenter input when the input exceeds the length allocated for it.

```
Please insert your Supplier ID
Enter 'X' to quit:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Please input a Supplier ID within 100 characters.
Please insert your Supplier ID
Enter 'X' to quit:
```

Output: Length Validation

- Line 420 - 431 will only run when supplier.txt is not empty.
- Line 426 - 431: Check file content to see if the user input item already exists, and close the file afterwards. If no (flag = 0), it exits the program because the entry does not appear in supplier.txt. If yes (flag = 1), it breaks the while loop and will go back to the main function that it is called from.

```
Please insert your Supplier ID
Enter 'X' to quit:test
Entry does not exist. Exiting program...
=====================================================================
Process finished with exit code 0
```

Output: Invalid Supplier ID Validation

- If supplier.txt is empty, it will just exit the program since there will be no point in checking the files.
- END: X or x

int distinctSupIDLineCount()

```
437  // To count how many distinct supID lines are in the file
438  int distinctSupIDLineCount() {
439      FILE *filePtr;
440      filePtr = fopen("supplier.txt", "r");
441      char buffer[MAX_PER_LINE_SIZE];                          // Buffer will store each line from the original file
442      int supFinder = 0;
443      char supID[SUP_ID_SIZE];                                 // To hold the value of generated supID
444
445      for(int i = 1; i <= lineCount( fileUsed: "supplier.txt"); ++i) {
446          while (fgets(buffer, MAX_PER_LINE_SIZE, filePtr) != NULL) { // Check by line (Doesn't repeat the line - Since supID entries are in ascending/incremental order)
447              // Constructing a Supplier ID
448              supID[0] = '\0';
449              strcat(supID, "SUP");
450              char placeholderNum[MAX_SIZE_OF_SUPPLIER_NO] = {0};
451              sprintf(placeholderNum, "%d", i);
452              strcat(supID, placeholderNum);
453
454              if ((strstr(buffer, supID)) != NULL) {           // Increases supFinder when the entry exists
455                  supFinder++;                                 // How many times supFinder occurred => Will be the distinct value
456                  break;
457              }
458          }
459      }
460      fclose(filePtr);
461      return supFinder;
462  }
```

- To count how many distinct supID lines are in the file.
- A file pointer is declared and “supplier.txt” will be opened for reading purposes.
- The buffer variable will store each line from the original file.
- The supFinder variable will be the distinct value of the supID occurrence.
- The supID variable holds the value of generated supIDs.
- A Supplier ID will be generated by concatenating the string “SUP” with a number that will increase by 1 through the for loop until the total number of lines in the file after being compared in strstr( ).
- Before generating new Supplier IDs, index 0 of supID is set to null to terminate the string regardless of the length of the array or memory.
- If the Supplier ID exits in supplier.txt, supFinder will increase by 1 and breaks the while loop so that the next new Supplier ID can be generated.
- After the for loop ends, supplier.txt will be closed and supFinder will be returned.

void partsInventoryCreation()

The main function for Section 1.

```
464     // Section 1 - Parts Inventory Creation
465     void partsInventoryCreation() {
466         // Assign an object to the structure
467         warehouse part;
468 
469         // Setting the file name as the respective warehouse.txt file
470         char *fileName = selectingWarehouse();
471 
472         // Picking a model and setting the warehouse name
473         char model[MODEL_SIZE];
474         if (strcmpi(fileName,"WBZ.txt") == 0) {
475             strcpy(model, "Blaze");
476             strcpy(part.warehouseChoice, "WBZ");
477         }
478         if (strcmpi(fileName,"WSL.txt") == 0) {
479             strcpy(model, "Silk");
480             strcpy(part.warehouseChoice, "WSL");
481         }
482         if (strcmpi(fileName,"WAR.txt") == 0) {
483             strcpy(model, "Armer");
484             strcpy(part.warehouseChoice, "WAR");
485         }
486 
487         printingSeparator( length: 69,  item: '-');
488         printf("\n");
```

1. Line 467: Assign an object (part) to the warehouse structure.
2. Line 470: Setting the file name as the respective warehouse.txt file by letting the user pick a model in the *selectingWarehouse function*.
3. Line 473 - 485: Picking a model and setting the warehouse name according to the file name set previously.

```
490         // To check if the inventory/part name already exists in the file chosen, if yes, user exits the program
491         part.partName[0] = '\0';
492         entryCheckForDuplicate(fileName,  item: "part", PART_NAME_SIZE, part.partName);
493         strcpy(part.partName, strupr(part.partName)); // Changing the part into UPPER CASE before placing it into the file
494
495         printingSeparator( length: 69,  item: '-');
496         printf("\n");
497
498         // Letting user insert the quantity for the new inventory
499         part.partQuantity = '\0';
500         printf("Please insert the quantity of part (%s) that you would like to register", part.partName);
501         printf("\nEnter negative numbers to quit:");
502         scanf("%d", &part.partQuantity);
503         if (part.partQuantity < 0) {
504             exitingProgram( status: 1);
505         }
506
507         printingSeparator( length: 69,  item: '-');
508         printf("\n");
```

4. Line 491 - 493: To check if the inventory (Part name) inputted by the user already exists in the file chosen in *entryCheckForDuplicate function*, if yes, it exits the program. Afterwards, the part name is changed into uppercase before placing it into the warehouse files using strcpy( ) and strupr( ).
5. strcpy( ) copies the string pointed by the source (Part name without uppercase) to the destination (Part name with upper case).
6. strupr( ) converts a given string to uppercase.
7. Line 499 - 505: Letting user insert the quantity for the new inventory and reads the input as part.partQuantity.
8. EXIT: Any negative numbers

```
510         // Count the number of lines in all 3 files to give a number for part ID
511         int counterForPart = 0;  // Total line count
512         counterForPart = lineCount( fileUsed: "WBZ.txt") + lineCount( fileUsed: "WSL.txt") + lineCount( fileUsed: "WAR.txt");
513
514         // Create a unique Part ID
515         part.partID[0] = '\0';                              // Clearing content - Terminate the string regardless of the length of the array or memory
516         char placeholderNum[MAX_SIZE_OF_PART_NO] = {0};
517         strcat(part.partID, part.warehouseChoice);
518         sprintf(placeholderNum, "%d", counterForPart + 1);      // Adding 1 to the ID so it starts with 1 instead of 0
519         strcat(part.partID, placeholderNum);
520         // NOTE: strcat() behavior is undefined if the destination array is not large enough for the contents of both src
521         //       and dest and the terminating null character
522         // Hence we need to leave extra (one) space for its null character
```

9. Line 511 & 512: Count the number of lines in all 3 files to give a number for the second part for part ID since a part ID consists of the warehouse name (3 letter code) and the sequence number.
10. Line 515 - 519: Create a unique Part ID by concatenating the warehouse and sequence number. The sequence number is increased by 1 so the partID sequence starts with 1 instead of 0. The number was placed into a placeholderNum variable to convert integer to string so that it can be concatenated.

11. Since strcat() behavior will be undefined if the destination array is not large enough for the contents of both source, destination, and the terminating null character, I have allocated a larger array size the avoid this error.

```
524         // Assign an object to the structure
525         supplier sup;
526 
527         // Getting the ID and name of the supplier
528         char option;
529         printf("Do you have an existing Supplier ID? [ Y (Yes) / N (No) / X (To quit) ]:");
530         scanf(" %c", &option);
531         strupr(&option);
532         seekToNextLine();
533         if (option == 'X') {
534             exitingProgram( status: 1);
535         }
```

12. Line 467: Assign an object (part) to the warehouse structure.
13. Line 528 - 532: Getting the ID and name of the supplier by prompting the user for an existing Supplier ID. If there are inputs other than “X”, “x”, “Y”, “y”, N” and “n”, the program will exit while indicating invalid input.
14. EXIT: X or x

```
536     else if (option == 'Y') {
537         if (lineCount( fileUsed: "supplier.txt") == 0) {
538             exitingProgram( status: 6); // Exit program -> No entries exist
539         } else {
540             entryCheckForExistenceSupID(sup.supID);
541             // Determining the status of stock for the respective part
542             FILE *fSection1PartExtra;
543             fSection1PartExtra = fopen(fileName, "a");
544             if (part.partQuantity < 10)
545                 fprintf(fSection1PartExtra, "%s|%s|%d|%s|LOW\n", part.partID, part.partName, part.partQuantity, sup.supID);
546             else
547                 fprintf(fSection1PartExtra, "%s|%s|%d|%s|NORMAL\n", part.partID, part.partName, part.partQuantity, sup.supID);
548             fclose(fSection1PartExtra);
549 
550             // Find the supName in supplier.txt using supID and put supplier information in "supplier.txt"
551             lineNumberFinderAndPrintResults( fileUsed: "supplier.txt", sup.supID,  choice: 4, part.partID);
552 
553             // Print Section 1 Results
554             printResults1(part.partID, model, part.partName, part.partQuantity, sup.supID);
555             exitingProgram( status: 1);
556         }
```

15. Line 536 - 556 will only run if the user inserts “Y” or “y” (Indicating that they have an existing Supplier ID). If supplier.txt is empty, it will exit the system while indicating that no entries exist.
16. Line 536 - 556: If supplier.txt is not empty, it will check if the Supplier ID that will be inserted by the user exists or not in *entryCheckForExistenceSupID function*. Then, it will determine the status of stock for the respective part. If the result variable is less than 10, the status part of the text to be

replaced will be “LOW”. Otherwise, it will be “NORMAL”. The record for the parts will then be appended into its respective warehouse file. Afterwards, it will find the supName in supplier.txt using supID and put supplier information into "supplier.txt" in the *lineNumberFinderAndPrintResults function*. The results for Section 1 will then be printed in *printResults1 function* and proceeds to exit the program.

```
557     } else if (option == 'N') {
558         // To check if the name already exists in "supplier.txt", if yes, exit program
559         entryCheckForDuplicate( fileUsed: "supplier.txt",  item: "name", SUP_NAME_SIZE, sup.supName);
560         strupr(sup.supName);
561 
562         // Count the number of lines in supplier.txt to give a number for part ID
563         int counterForSupplier;                                 // Total line count
564         counterForSupplier = distinctSupIDLineCount();
565 
566         // Create a unique Supplier ID
567         sup.supID[0] = '\0';                                    // Clearing content - Terminate the string regardless of the length of the array or memory
568         char placeholderNum2[MAX_SIZE_OF_SUPPLIER_NO] = {0};    // To store number so it could be considered as strings to be used
569         strcat(sup.supID, "SUP");
570         sprintf(placeholderNum2, "%d", counterForSupplier + 1);    // Adding 1 to the ID so it starts with 1 instead of 0
571         strcat(sup.supID, placeholderNum2);
572     } else {
573         exitingProgram( status: 3); // Exit program -> Invalid input
574     }
```

17. Line 557 - 571 will only run if the user inserts “N” or “n” (Indicating that they do not have an existing Supplier ID).
18. Line 557 - 571: It will check if the supplier name inputted by the user already exists in "supplier.txt" through the *entryCheckForDuplicate function*. If yes, it will exit the program. Then, it counts the number of lines in supplier.txt to give a number for part ID using the *distinctSupIDLineCount* function. A unique Supplier ID will then be created by concatenating “SUP” with a number acquired previously.

```
623         // Put parts information in its respective warehouse ".txt"
624         FILE *fSection1Part;
625         fSection1Part = fopen(fileName, "a");
626 
627         // Determining the status of stock for the respective part
628         if (part.partQuantity < 10)
629             fprintf(fSection1Part, "%s|%s|%d|%s|LOW\n", part.partID, part.partName, part.partQuantity, sup.supID);
630         else
631             fprintf(fSection1Part, "%s|%s|%d|%s|NORMAL\n", part.partID, part.partName, part.partQuantity, sup.supID);
632         fclose(fSection1Part);
633 
634         // Put supplier information in "supplier.txt"
635         FILE *fSection1Sup;
636         fSection1Sup = fopen("supplier.txt", "a");
637         fprintf(fSection1Sup, "%s|%s|%s\n", sup.supID, sup.supName, part.partID);
638         fclose(fSection1Sup);
639 
640         // Print Section 1 Results
641         printResults1(part.partID, model, part.partName, part.partQuantity, sup.supID);
642     }
```

19. Line 624 - 632: Put parts information in its respective warehouse file after determining the status of stock for the respective part (Similar to the process mentioned earlier).

20. Line 635 - 638: Putting supplier information in "supplier.txt" via append mode.
21. Line 641: Prints the results for Section 1 through the *printResults1 function*.

Output of Section 1 (User Does Not Have an Existing Supplier ID)

```
Pick a model by entering the corresponding number:
        1 - Blaze (BZ)
        2 - Silk (SL)
        3 - Armer (AR)
Enter negative numbers to quit:1
------------------------------------------------------------------
Please insert the part that you want to register
Enter 'X' to quit:engine
------------------------------------------------------------------
Please insert the quantity of part (ENGINE) that you would like to register
Enter negative numbers to quit:45
------------------------------------------------------------------
Do you have an existing Supplier ID? [ Y (Yes) / N (No) / X (To quit) ]:n
Please insert the name that you want to register

Enter 'X' to quit:carsome beep beep
------------------------------------------------------------------
                    Inventory Creation Details
------------------------------------------------------------------
You have successfully registered WBZ17.
        Model Chosen      = Blaze
        Part ID           = WBZ17
        Part Name         = ENGINE
        Part Quantity     = 45
        Supplier ID       = SUP11
------------------------------------------------------------------
Details are successfully printed.
```

File Output of Section 1 (User Does Not Have an Existing Supplier ID)

```
SUP11|CARSOME BEEP BEEP|WBZ17
```

Output In supplier.txt

```
WBZ17|ENGINE|45|SUP11|NORMAL
```

Output In WBZ.txt

Output of Section 1 (User Have an Existing Supplier ID)

```
Pick a model by entering the corresponding number:
        1 - Blaze (BZ)
        2 - Silk (SL)
        3 - Armer (AR)
Enter negative numbers to quit:2
------------------------------------------------------------------------
Please insert the part that you want to register
Enter 'X' to quit:hood
------------------------------------------------------------------------
Please insert the quantity of part (HOOD) that you would like to register
Enter negative numbers to quit:48
------------------------------------------------------------------------
Do you have an existing Supplier ID? [ Y (Yes) / N (No) / X (To quit) ]:y
Please insert your Supplier ID
Enter 'X' to quit:sup8
------------------------------------------------------------------------
                    Inventory Creation Details
------------------------------------------------------------------------
You have successfully registered WSL18.
        Model Chosen      = Silk
        Part ID           = WSL18
        Part Name         = HOOD
        Part Quantity     = 48
        Supplier ID       = SUP8
------------------------------------------------------------------------
Details are successfully printed.
Thank you for using this system :D
```

File Output of Section 1 (User Have an Existing Supplier ID)

```
SUP8|JUSTIN|WSL18
```

Output In supplier.txt

```
WSL18|HOOD|48|SUP8|NORMAL
```

Output In WBZ.txt

==========

SECTION 2 FUNCTIONS

Functions that are only used in Section 2 - Parts Inventory Update.

==========

void partsInventoryUpdate()

The main function for Section 2.

```
646     // Section 2 - Parts Inventory Update
647     void partsInventoryUpdate() {
648         // Assign an object to the structure
649         warehouse part;
650 
651         // Check if Part ID exists and return its respective filename
652         char *fileName = entryCheckForExistencePartID(part.partID);
653 
654         // Find the line in warehouse.txt using partID, update stock, and print results
655         lineNumberFinderAndPrintResults(fileName, part.partID,  choice: 3,  extraPartForSection1: "Empty");
656     }
```

1. Assigns an object (part) to the warehouse structure.
2. Check if Part ID inserted by the user exists and return its respective filename in the *entryCheckForExistencePartID function*.
3. Find the line in warehouse.txt using the partID, update the stock, and print the results in the *lineNumberFinderAndPrintResults function*.

Output of Section 2 (Increase Stock Quantity)

```
Please insert your Part ID
Enter 'X' to quit:wsl9

------------------------------------------------------------------
                          Part Details
------------------------------------------------------------------
        Part ID           = WSL9
        Part Name         = TAIL LIGHTS
        Part Quantity     = 5
        Part Status       = LOW
        Supplier ID       = SUP5
------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - Increase stock quantity
        2 - Decrease stock quantity
Enter negative numbers to quit:1
Please insert the stock amount to be added:50

Details updated. Returning to main menu...
```

File Output of Section 2 (Increase Stock Quantity)

```
WSL9|TAIL LIGHTS|55|SUP5|NORMAL
```

Output In WSL.txt

Output of Section 2 (Decrease Stock Quantity)

```
Please insert your Part ID
Enter 'X' to quit:wsl18

------------------------------------------------------------------------
                            Part Details
------------------------------------------------------------------------
        Part ID           = WSL18
        Part Name         = HOOD
        Part Quantity     = 48
        Part Status       = NORMAL
        Supplier ID       = SUP8
------------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - Increase stock quantity
        2 - Decrease stock quantity
Enter negative numbers to quit:2
Please insert the stock amount to be subtracted:38

Details updated. Returning to main menu...
```

File Output of Section 2 (Decrease Stock Quantity)

```
WSL18|HOOD|10|SUP8|NORMAL
```

Output In WSL.txt

==========

SECTION 3 FUNCTIONS

Functions that are only used in Section 3 - Parts Inventory Tracking.

==========

void partsInventoryTracking()

The main function for Section 3.

```
660     // Section 3 - Parts Inventory Tracking
661     void partsInventoryTracking() {
662         // Setting the file name as the respective warehouse.txt file
663         char *fileName = selectingWarehouse();
664 
665         FILE *filePtr;
666         char storeLine[MAX_PER_LINE_SIZE];
667         filePtr = fopen(fileName, "r");
668         int LOWStatusCounter = 0;
669 
670         // Select from menu
671         int choice;
672         printingSeparator( length: 69,  item: '-');
673         printf("\nPick an operation by entering the corresponding number:");
674         printf("\n\t1 - Print all parts in ascending order (According to Part ID)\n\t2 - Print parts that has < 10 stock");
675         while (1) {
676             printf("\nEnter negative numbers to quit:");
677             scanf("%d", &choice);
678             seekToNextLine();
```

1. Line 663: Setting the file name as the respective warehouse.txt file in the *selectingWarehouse function*.
2. Line 665 - 668: Creates a file pointer and open the file provided in the fileName variable as read mode. It also declares storeLine variable which will be used to store each line in a file. The LOWStatusCounter is to count the occurrence of “LOW” status in the file.
3. Line 671 - 678: Prints a menu (To print all parts in ascending order (According to Part ID) or to print parts that have less than 10 stock) and reads the input of the user and store it in a variable called choice.

```
679             if (lineCount(fileName) == 0) {
680                 printf("This file has no records yet.\n");
681                 break;
682             }
```

4. Line 679 - 682: If the fileName variable is empty, it will break from the while loop indicating that the file has no records yet.

```
683         if (choice > 0 && choice < 3) {
684             printf("\n");
685             printingSeparator( length: 69,  item: '-');
686             printf("\n");
687             messageCentered( message: "Parts Inventory Details");
688             printf("\n");
689             printingSeparator( length: 69,  item: '-');
690             printf("\n");
691             if (choice == 1) {
692                 while (fgets(storeLine, MAX_PER_LINE_SIZE, filePtr) != NULL) {   // Check by line
693                     printf("%s\n", storeLine);
694                 }
695             } else {    // if (choice == 2)
696                 while (fgets(storeLine, MAX_PER_LINE_SIZE, filePtr) != NULL) {
697                     if ((strstr(storeLine, "LOW")) != NULL) {
698                         LOWStatusCounter++;
699                         printf("%s\n", storeLine);
700                     }
701                 }
702                 if (LOWStatusCounter == 0) {
703                     printf("\nNo parts are less than 10 in %s.\n", fileName);
704                 }
705             }
706             fclose(filePtr);
707             break;
```

5. Line 684 - 690: Prints the header for the output of choice 1 and 2.
6. Line 691 - 694: If choice is 1, it prints all the records in the file by going through the file via a while loop. This can be done directly since the records in the warehouse file are in ascending order no matter what.
7. Line 695 - 705: If choice is 2, it prints all the records in the file that contains “LOW” status by going through a file loop and strstr( ). When a file contains “LOW” status, the LOWStatusCounter variable will increase as well. After the while loop ends, if LOWStatusCounter remains 0 (Meaning that all the parts in the file has a “NORMAL” status), it will print that there were no parts that have less than 10 stock.
8. Line 706 - 707: Closes the file to avoid further modification and break from the while loop.

```
708             } else if (choice == 0 || choice > 2) {
709                 // Validating input
710                 printf("Please choose a number from 1 to 2 only.\n");
711             } else {
712                 // If input is a negative number, exit program
713                 exitingProgram( status: 1);
714             }
715         }
716     }
```

9. Line 708 - 710: If the user input for the choice variable is 0 or more than 2 (Values that were not shown in the menu), it will ask the user to reenter input.
10. EXIT: Any negative numbers

Output of Section 3 (All Parts In Ascending Order)

```
Pick a model by entering the corresponding number:
        1 - Blaze (BZ)
        2 - Silk (SL)
        3 - Armer (AR)
Enter negative numbers to quit:3
------------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - Print all parts in ascending order (According to Part ID)
        2 - Print parts that has < 10 stock
Enter negative numbers to quit:1

------------------------------------------------------------------------
                        Parts Inventory Details
------------------------------------------------------------------------
WAR1|HOOD|10|SUP1|NORMAL

WAR3|BATTERY|56|SUP3|NORMAL

WAR4|SUSPENSION|56|SUP2|NORMAL

WAR8|BUMPER|3|SUP7|LOW

WAR10|ENGINE|10|SUP1|NORMAL

WAR16|BRAKE|6|SUP10|LOW
```

Output of Section 3 (Parts That Has Less Than 10 In Stock)

```
Pick a model by entering the corresponding number:
        1 - Blaze (BZ)
        2 - Silk (SL)
        3 - Armer (AR)
Enter negative numbers to quit:3
------------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - Print all parts in ascending order (According to Part ID)
        2 - Print parts that has < 10 stock
Enter negative numbers to quit:2

------------------------------------------------------------------------
                        Parts Inventory Details
------------------------------------------------------------------------
WAR8|BUMPER|3|SUP7|LOW

WAR16|BRAKE|6|SUP10|LOW
```

Output of Section 3 (Warehouse That Does Not Have Parts Less Than 10 In Stock)

```
Pick a model by entering the corresponding number:
        1 - Blaze (BZ)
        2 - Silk (SL)
        3 - Armer (AR)
Enter negative numbers to quit:2
------------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - Print all parts in ascending order (According to Part ID)
        2 - Print parts that has < 10 stock
Enter negative numbers to quit:2

------------------------------------------------------------------------
                        Parts Inventory Details
------------------------------------------------------------------------

No parts are less than 10 in WSL.txt.
```

==========

SECTION 4 FUNCTIONS

Functions that are only used in Section 4 - Search Records & Supplier Details.

==========

void searchRecordsAndSupplierDetails()

The main function for Section 4.

```
720     // Section 4 - Search Records & Supplier Details
721     void searchRecordsAndSupplierDetails() {
722         // Assign an object to the structure
723         warehouse part;
724 
725         // Check if Part ID exists and return its respective filename
726         char *fileName = entryCheckForExistencePartID(part.partID);
```

1. Line 723: Assigns an object (part) to the warehouse structure.
2. Line 726: Check if Part ID inserted by the user exists and return its respective filename in the *entryCheckForExistencePartID function*.

```
728         // Select from menu
729         int choice;
730         printingSeparator( length: 69,  item: '-');
731         printf("\nPick an operation by entering the corresponding number:");
732         printf("\n\t1 - View Part Details\n\t2 - View Supplier Details");
733         while (1) {
734             printf("\nEnter negative numbers to quit:");
735             scanf("%d", &choice);
736             seekToNextLine();
737             if (choice > 0 && choice < 3) {
738                 if (choice == 1) {
739                     // Find the line in warehouse.txt using partID and print results
740                     lineNumberFinderAndPrintResults(fileName, part.partID,  choice: 1,  extraPartForSection1: "Empty");
741                 } else {     // if (choice == 2)
742                     // Find the line in supplier.txt using partID and print results
743                     lineNumberFinderAndPrintResults( fileUsed: "supplier.txt", part.partID,  choice: 2,  extraPartForSection1: "Empty");
744                 }
745                 break;
746             } else if (choice == 0 || choice > 2) {
747                 // Validating input
748                 printf("Please choose a number from 1 to 2 only.\n");
749             } else {
750                 // If input is a negative number, exit program
751                 exitingProgram( status: 1);
752             }
753         }
754     }
```

3. Line 729 - 736: Prints a menu (To view part or supplier details) and reads the input of the user and store it in a variable called choice.

4. Line 737 - 745: If choice is 1, it will find the line in the warehouse file using partID and print the results for section 4. If choice is 2, it will find the line in supplier.txt using partID and print the results for section 4. Afterwards, it will break from the while loop.
5. Line 746 - 748: If the user input for the choice variable is 0 or more than 2 (Values that were not shown in the menu), it will ask the user to reenter input.
6. EXIT: Any negative numbers

Output of Section 4 (View Part Deatails)

```
Please insert your Part ID
Enter 'X' to quit:wsl2
------------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - View Part Details
        2 - View Supplier Details
Enter negative numbers to quit:1

------------------------------------------------------------------------
                            Part Details
------------------------------------------------------------------------
        Part ID           = WSL2
        Part Name         = WINDSHIELD WIPER
        Part Quantity     = 69
        Part Status       = NORMAL
        Supplier ID       = SUP2
------------------------------------------------------------------------
```

Output of Section 4 (View Supplier Deatails)

```
Please insert your Part ID
Enter 'X' to quit:wsl2
------------------------------------------------------------------------
Pick an operation by entering the corresponding number:
        1 - View Part Details
        2 - View Supplier Details
Enter negative numbers to quit:2

------------------------------------------------------------------------
                          Supplier Details
------------------------------------------------------------------------
        Part ID           = WSL2
        Supplier ID       = SUP2
        Supplier Name     = IAN
------------------------------------------------------------------------
```

==========

MAIN FUNCTIONS

Functions that are only used in the main code.

==========

void fileCheck()

```
// Generate the necessary files
void fileCheck() {
    FILE *fptr;
    char files[4][FILENAME_MAX] = {"WBZ.txt", "WSL.txt", "WAR.txt", "supplier.txt"}; // Loop for each file
    for(int i = 0; i < 4; i++) {
        // Returns 0 if the files exists
        if(access(files[i], F_OK) != 0) {
            fptr = fopen(files[i], "w");
            fclose(fptr);
        }
    }
}
```

- To generate the necessary files during the start of the program if the files do not exist.
- The names of the 4 required files are placed in an array called files.
- access( ) determines if a file can be accessed or not.
- F_OK tests whether the file exists in the same directory.
- The 4 files go on a loop to check one by one if they exist.
- If the current file on loop does not exist in the same directory as main.c, access will not return 0 and the file will be generated.

---

void printMenu()

```
// Print menu
void printMenu() {
    printf("\n");
    printingSeparator( length: 69,  item: '=');
    printf("\n");
    messageCentered( message: "Welcome to the Automobile Parts Management System");
    printf("\n");
    printingSeparator( length: 69,  item: '=');
    printf("\n\t1. Parts Inventory Creation In Warehouse\n\t2. Parts Inventory Update\n\t3. Parts Inventory Tracking");
    printf("\n\t4. Search Records & Supplier Details\n\t5. Exit");
}
```

To print the main menu for the system.

---

void menuSelection()

```
783     // Selecting operation from menu
784     void menuSelection() {
785         int choice;
786         while (1) {
787             // Check if the file needed exists, if not then create new ones
788             fileCheck();
789 
790             // Print menu
791             printMenu();
792             printf("\nPlease choose any operation from the given options:");
793             scanf("%d", &choice);
794             seekToNextLine();
795 
796             // Input validation
797             while ((choice <= 0) || (choice > 5)) {
798                 printf("Please choose a number from 1 to 5 only.");
799                 printf("\n\nPlease choose any operation from the given options:");
800                 scanf("%d", &choice);
801                 seekToNextLine();
802             }
```

- Line 788: Check if the file needed exists, if not then create new ones through *fileCheck function*.
- Line 791 - 794: Prints the main menu and reads the input of the user and store it in a variable called choice.
- Line 797 - 802: If the user input for the choice variable is 0 or more than 5 (Values that were not shown in the menu), it will ask the user to reenter input.

```
804             switch (choice) {
805                 case 1:
806                     // Parts Inventory Creation In Warehouse
807                     printingSeparator( length: 69,  item: '=');
808                     partsInventoryCreation();
809                     break;
810                 case 2:
811                     // Parts Inventory Update
812                     printingSeparator( length: 69,  item: '=');
813                     if (lineCount( fileUsed: "supplier.txt") > 0)
814                         partsInventoryUpdate();
815                     else
816                         printf("No record exists.\nPlease register in 'Section 1 - Parts Inventory Creation In Warehouse' before proceeding to this section.\n");
817                     break;
818                 case 3:
819                     // Parts Inventory Tracking
820                     printingSeparator( length: 69,  item: '=');
821                     if (lineCount( fileUsed: "supplier.txt") > 0)
822                         partsInventoryTracking();
823                     else
824                         printf("No record exists.\nPlease register in 'Section 1 - Parts Inventory Creation In Warehouse' before proceeding to this section.\n");
825                     break;
826                 case 4:
827                     // Search Records & Supplier Details
828                     printingSeparator( length: 69,  item: '=');
829                     if (lineCount( fileUsed: "supplier.txt") > 0)
830                         searchRecordsAndSupplierDetails();
831                     else
```

```
832                         printf("No record exists.\nPlease register in 'Section 1 - Parts Inventory Creation In Warehouse' before proceeding to this section.\n");
833                     break;
834                 default:
835                     // End the program
836                     printf("Thank you for using this system :D\n");
837                     printingSeparator( length: 69,  item: '=');
838                     exit(0);
839             }
840         }
841     }
```

- Line 804 - 839: A switch case that directs the user to different functions according to the choice variable value:
    - 1: Section 1 - Parts Inventory Creation In Warehouse
    - 2: Section 2 - Parts Inventory Update
    - 3: Section 3 - Parts Inventory Tracking
    - 4: Section 4 - Search Records and Supplier Details
- END: 5

```
=====================================================================
          Welcome to the Automobile Parts Management System
=====================================================================
        1. Parts Inventory Creation In Warehouse
        2. Parts Inventory Update
        3. Parts Inventory Tracking
        4. Search Records & Supplier Details
        5. Exit
Please choose any operation from the given options:5
Thank you for using this system :D
=====================================================================
Process finished with exit code 0
```

Sample Output

int main()

```
843 ▶   int main() {
844         menuSelection();
845         return 0;
846     }
```

Calls the main menu selection.

==========

MAIN CODE

Calls main( ) and includes preprocessor directives, define statements, as well as structure definitions.

==========

```
4   #include <stdio.h>
5   #include <stdlib.h>
6   #include <string.h>
7   #include <unistd.h>
8   #include <stdbool.h>
```

Preprocessor directives

```
10  #define FILENAME_SIZE 100
11  #define MODEL_SIZE 100
12  #define WAREHOUSE_CHOICE_SIZE 100
13  #define PART_ID_SIZE 100
14  #define PART_NAME_SIZE 100
15  #define SUP_ID_SIZE 100
16  #define SUP_NAME_SIZE 100
17  #define MAX_SIZE_OF_SUPPLIER_NO 6               // 5 digits max
18  #define MAX_SIZE_OF_PART_NO 6                   // 5 digits max
19  #define MAX_PER_LINE_SIZE 200                   // Maximum characters per line in a file
20  #define MAX_SIZE_OF_QUANTITY 6                  // 5 digits max
```

Define statements that are mostly for the sizes of character arrays (Strings)

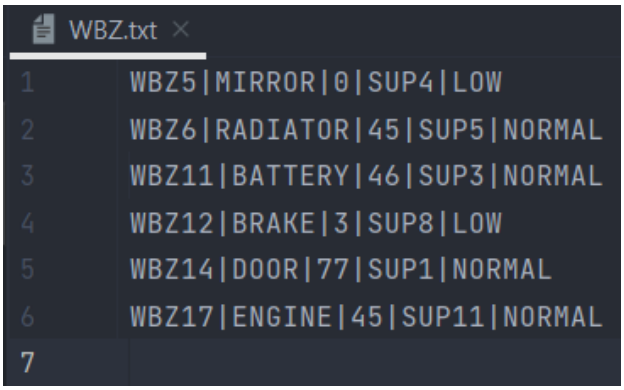
```c
typedef struct Warehouse {
    char warehouseChoice[WAREHOUSE_CHOICE_SIZE];
    char partID[PART_ID_SIZE];
    char partName[PART_NAME_SIZE];
    int partQuantity;
} warehouse;

typedef struct Supplier {
    char supID[SUP_ID_SIZE];
    char supName[SUP_NAME_SIZE];
} supplier;
```

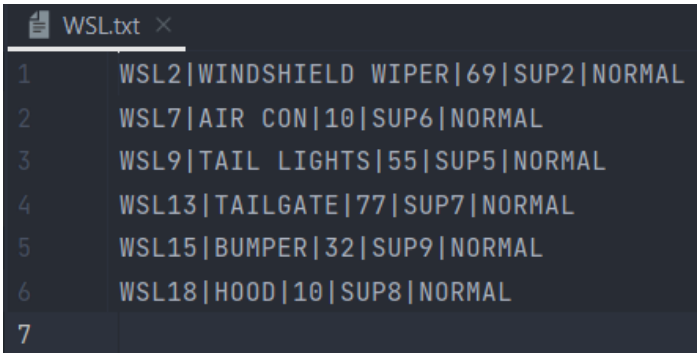
Structure definition for Warehouse and Supplier

CONTENT IN TEXT FILES

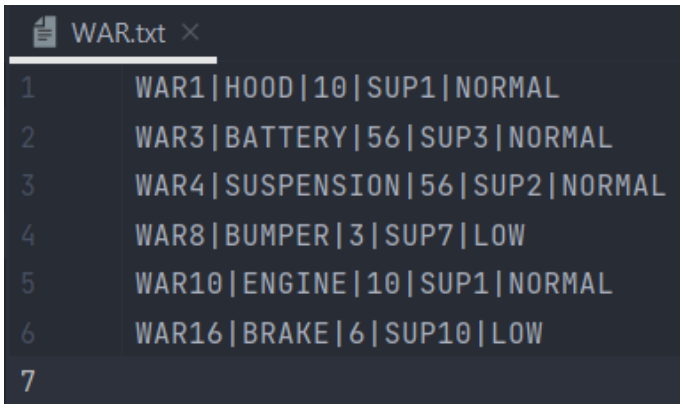
WBZ.txt



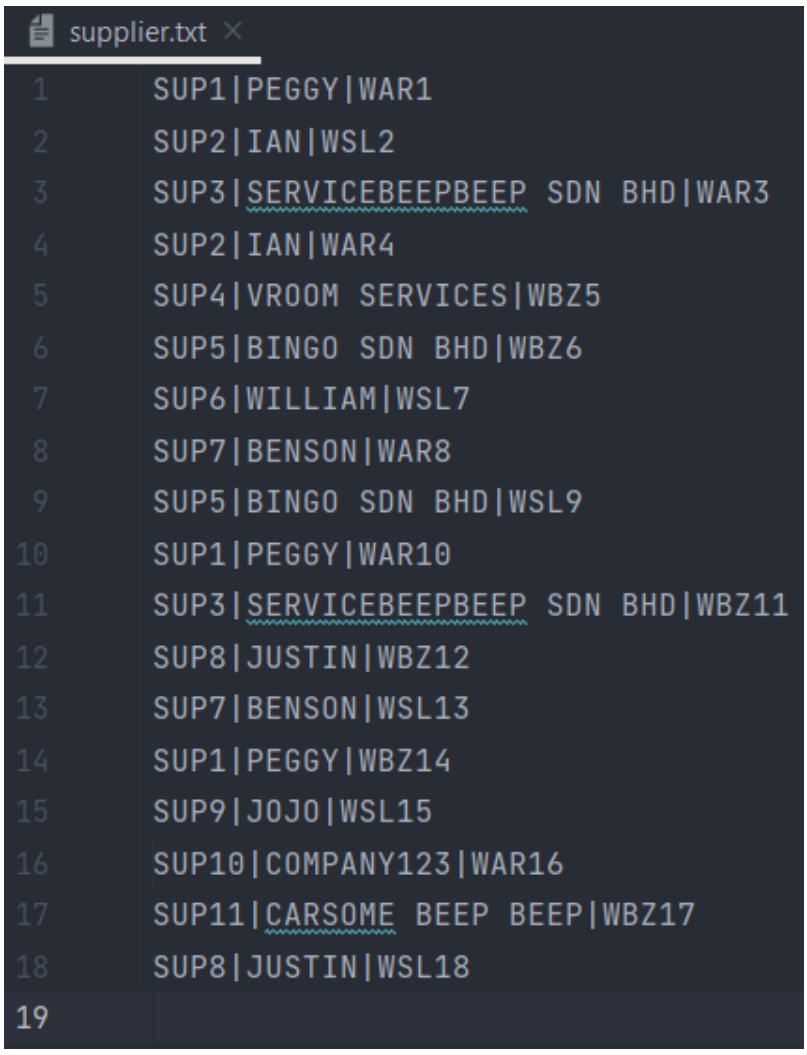
WSL.txt



WAR.txt



supplier.txt



CONCLUSION

This C programme makes use of modular programming approaches and is divided into four sections, each with a distinct number of functions, as well as 12 generic functions, four main functions, and the main code. This is done in order to reuse codes, which improves readability. Furthermore, the programme is menu-driven, which gives users, particularly digital immigrants, a better user experience. For improved management and readability, good programming standards such as variable name conventions in camel casing, comments, and indentation are used.

I have learnt a lot about programming practices and the usage of data structures during the assignment, and I am very glad that I got the chance to solve problems like these because it puts my practical skills to the test.

REFERENCES

IBM. (2021). *access() — Determine whether a file can be accessed*. IBM. https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-access-determine-whether-file-can-be-accessed

Kumar, A. (2017). *Predefined string functions of C in string.h library*. CodesDope. https://www.codesdope.com/blog/article/predefined-string-functions-of-c-in-stringh-librar/