INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT018-3-1-ICP

INTRODUCTION TO C PROGRAMMING

APD1F2106CS

HAND OUT DATE: 10 JANUARY 2022

HAND IN DATE: 25 FEBRUARY 2022

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1. Submit your assignment online in Moodle unless advised otherwise
2. Late submission will be awarded zero(0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

Name : Yam Chen Xi

TP Number : TP061635

Table of Contents

1.0 Introduction and Assumptions 1
2.0 Design of the Program 2
2.1 Pseudocode 2
2.2 Flowchart 16
3.0 Program Source Code 49
3.1 Header file 49
main.h 49
menu.h 49
fileHandling.h 49
partsInventoryCreation.h 50
partsSupplier.h 50
partsInventoryUpdate.h 51
partsInventoryTracking.h 51
searchingFunctionalities.h 51
3.2 Source File 52
main.c 52
menu.c 53
fileHandling.c 54
partsInventoryCreation.c 56
partsSupplier.c 58
partsInventoryUpdate.c 59
partsInventoryTracking.c 62
searchingFunctionalities.c 66
4.0 Program Sample Outputs 69
4.1 Main Menu 69
4.2 Parts Inventory Creation 70
4.3 Parts Supplier Creation 71
4.4 Parts Inventory Update 72
4.5 Parts Inventory Tracking 74
4.6 Searching Functionalities 76
4.7 Exit 78
5.0 Text Files 79

5.1 Supplier File - Supplier.txt 79
5.2 Warehouse Inventory File 80
5.2.1 WBZinventory.txt 80
5.2.2 WSL inventory.txt 80
5.2.3 WARinventory.txt 80
6.0 Conclusion 81

1.0 Introduction and Assumptions

Creating an automobile parts inventory management system is the main objective of this assignment to manage automobile parts in all the warehouses. Flowchart and pseudocode are designed to convert problem statements into programmable solutions before C programming is involved. The system has 4 features namely the Parts Inventory Creation in Warehouses, Parts Inventory Update, Parts Inventory Tracking and Searching Functionalities. All the features are then incorporated into a system that allows the user to easily navigate between functions using the menu.

2.0 Design of the Program

2.1 Pseudocode

main.h

```
DEFINE _CRT_SECURE_NO_WARNINGS
INCLUDE <stdio.h>
INCLUDE <stdlib.h>
```

main.c

```
ROGRAM automobilePartsInventoryManagementSystem
BEGIN

INCLUDE "main.h"
INCLUDE "menu.h"P

     char cont
     Print "Welcome to Automobile Parts Inventory Management System\n "
     DOWHILE (cont != "x") or (cont != "X")
          CALL menu()
          print "\nPress <enter> to continue OR 'X' to end: "
          read cont
     ENDDO
     Print "Thank you and have a great day! \n"
     RETURN 0
END
```

menu.h

```
IFNDEF menu_header
DEFINE menu_header

void menu()

ENDIF
```

menu.c

```
INCLUDE "partsInventoryCreation.h"
INCLUDE "partsSupplier.h"
INCLUDE "partsInventoryTracking.h"
INCLUDE "partsInventoryUpdate.h"
INCLDUE "searchingFunctionalities.h"

FUNCTION void menu()
     DECLARE char option
     Print "1. Parts Inventory Creation in Warehouses\n2. Parts Supplier Creation\n3. Parts
Inventory Update\n4. Parts Inventory Tracking\n5. Searching Part Inventory\n6. Searching
Functionalities\n6. Exit\n"
     DECLARE int option
     print "Menu Option: "
     read option
     CASE OF option
          = 1:
          CALL partsInventoryCreation()
          BREAK
          = 2:
          CALL partsSupplierCreation()
          BREAK
          = 3:
          CALL partsInventoryUpdate()
          BREAK
          = 4:
          CALL partsInventoryTracking()
          BREAK
          = 5:
          CALL searchingFunctionalities()
          BREAK
          = 6:
          BREAK
     OTHERWISE
          Print "Invalid menu selection, select only from (1) to (6)."
     ENDCASE
ENDFUNCTION
```

fileHandling.h

```
INCLUDE "partsSupplier.h"
INCLUDE "partsInventoryCreation.h"

FILE* openWarehouseFile(char* warehouseCode)
int readWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData)
int addWarehouseRecord(FILE* fileHandle, partsInventory* partData)
int updateWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData)
int getWarehouseRecordCount(FILE* fileHandle)
void closeWarehouseFile(FILE* fileHandle)

FILE* openSupplierFile()
int readSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData)
int addSupplierRecord(FILE* fileHandle, partsSupplier* supplierData)
int updateSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData)
int getSupplierRecordCount(FILE* fileHandle)
void closeSupplierFile(FILE* fileHandle)
```

fileHandling.c

```
INCLUDE "main.h"
INCLUDE "partsInventoryCreation.h"
INCLUDE "partsSupplier.h"

FUNCTION FILE* openWarehouseFile(char* warehouseCode)
     DECLARE char fileName
     FILE* fileHandle = 0

     stringprint(fileName, "%sinventory.txt", warehouseCode)
     fileHandle = OPENFILE fileName for READ

     IF (0 == fileHandle) THEN
          fileHandle = OPENFILE fileName for WRITE
     ENDIF
     RETURN fileHandle
ENDFUNCTION


FUNCTION int readWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData)
     IF (0 > fseek(fileHandle, recordID * sizeof(partsInventory), SEEK_SET)) THEN
          RETURN -1
     ENDIF
     READFILE(partData, sizeof(partsInventory), 1, fileHandle)
     RETURN 0
ENDFUNCTION


FUNCTION int addWarehouseRecord(FILE* fileHandle, partsInventory* partData)
     DECLARE int s = 0
     fseek(fileHandle, 0L, SEEK_END)
     s = FILEWRITE(partData, sizeof(partsInventory), 1, fileHandle)
     RETURN (sizeof(partsInventory) == s)
ENDFUNCTION
```

```
FUNCTION int updateWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData)
     DECLARE int writeSize = 0
     IF (0 > fseek(fileHandle, recordID * sizeof(partsInventory), SEEK_SET))
          RETURN -1
     ENDIF
     writeSize = FILEWRITE (partData, sizeof(partsInventory), 1, fileHandle))
     fflush(fileHandle)
     RETURN 0
ENDFUNCTION


FUNCTION int getWarehouseRecordCount(FILE* fileHandle)
     fseek(fileHandle, 0L, SEEK_END)
     RETURN ftell(fileHandle) / sizeof(partsInventory))
ENDFUNCTION


FUNCTION void closeWarehouseFile(FILE* fileHandle)
     CLOSEFILE(fileHandle)
ENDFUNCTION
FUNCTION openSupplierFile(){
      DECLARE char fileName[128] = { 0 }
      FILE* fileHandle = 0
      fileHandle = OPENFILE "Supplier.txt" for append
      RETURN fileHandle
ENDFUNCTION
FUNCTION int readSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData)
     IF (0 > fseek(fileHandle, recordID * sizeof(partsSupplier), SEEK_SET))
          RETURN -1
     ENDIF
     READFILE(supplierData, sizeof(partsSupplier), 1, fileHandle)
     RETURN 0
ENDFUNCTION

FUNCTION int addSupplierRecord(FILE* fileHandle, partsSupplier* supplierData)
     DECLARE int s = 0
     fseek(fileHandle, 0L, SEEK_END)
     s = WRITEFILE(supplierData, sizeof(partsSupplier), 1, fileHandle)
     RETURN (sizeof(partsSupplier) == s)
ENDFUNCTION

FUNCTION int updateSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData)
     IF (0 > fseek(fileHandle, recordID * sizeof(partsSupplier), SEEK_SET))
          RETURN -1
     ENDIF
     RETURN (sizeof(partsSupplier) == fwrite(supplierData, sizeof(partsSupplier), 1, fileHandle))
ENDFUNCTION
 FUNCTION int getSupplierRecordCount(FILE* fileHandle)
      fseek(fileHandle, 0L, SEEK_END)
      RETURN (ftell(fileHandle) / sizeof(partsSupplier))
 ENDFUNCTION

 FUNCTION void closeSupplierFile(FILE* fileHandle)
      fclose(fileHandle)
 ENDFUNCTION
```

partInventoryCreation.h

```
IFNDEF partInventoryCreation_Header
DEFINE partInventoryCreation_Header

STRUCT partsInventory
     char warehouseCode[4], modelCode[3], sectionCode[9], partID[9], partName[16]
     int quantity
ENDSTRUCT

void partsInventoryCreation()
void printInventoryData(partsInventory* inv)
int isInteger(char* checkData)

ENDIF
```

partsInventoryCreation.c

```
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"


FUNCTION void printInventoryData(partsInventory* inv)
     Print "warehouseCode\t: %s\n", inv->warehouseCode
     Print "modelCode\t: %s\n", inv->modelCode
     Print "partID\t\t: %s\n", inv->partID
     Print "partName\t: %s\n", inv->partName
     Print "quantity\t: %d\n", inv->quantity
ENDFUNCTION


FUNCTION int isInteger(char* checkData)
     IF (0 == checkData) THEN
          RETURN 0
     ENDIF
     IF (checkData[0] == '0' && strlen(checkData) == 1) THEN
          RETURN 1
     ENDIF
     IF (0 == atoi(checkData)) THEN
          RETURN 0
     ENDIF
     RETURN 1
ENDFUNCTION
FUNCTION void partsInventoryCreation()
     FILE* warehouseFile
     partsInventory S1
     DECLARE int latestRecord = 0
     DECLARE int quantity = 0
     DECLARE int convertquantity
     DECLARE char warehouseCode[5]
```

```
     print "1. Parts Inventory Creation in Warehouses\n"
reinputwarehouse:
     memset(S1, 0 , sizeof(partsInventory))
     print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nEnter Warehouse Code: "
     Read S1.warehouseCode

     IF ((S1.warehouseCode = "1") OR (S1.warehouseCode = "WBZ")) THEN
          S1.modelCode = "BZ"
          S1.warehouseCode = "WBZ"
     ELSE IF ((S1.warehouseCode = "2") OR (S1.warehouseCode = "WSL"))
          S1.modelCode = "SL"
          S1.warehouseCode = "WSL"
     ELSE IF ((S1.warehouseCode = "3") OR (S1.warehouseCode = "WAR"))
          S1.modelCode = "AR"
          S1.warehouseCode = "WAR"
     ELSE
          Printf "\nInvalid warehouse code. Please try again!\n"
          GOTO reinputwarehouse
     ENDIF
      warehouseFile = CALL openWarehouseFile(S1.warehouseCode);
      latestRecord = getWarehouseRecordCount(warehouseFile)
      latestRecord = latestRecord + 1
      stringprint(S1.partID, "%s%03d", warehouseCode, latestRecord)
      Print "\nPart ID\t: %s%03d\n\n", warehouseCode, latestRecord

      Print "\nEnter Section Code: "
      Read S1.sectionCode
      Print "Enter Part Name\t: "
      Read S1.partName

 reinputquantity:
      Print "Enter Quantity\t: "
      Read S1.quantity

      IF (NOT CALL isInteger(S1.quantity)) THEN
           Print "\nInput invalid, please try again with number\n"
           StrCopy(S1.quantity, "")
           GOTO reinputquantity
      ENDIF
      S1.linefeed = '\n'

      latestRecord = CALL addWarehouseRecord(warehouseFile, &S1)
      Print "\n\Record successfully created!\n"

      latestRecord = CALLgetWarehouseRecordCount(warehouseFile)
      Print "Current record count: %d\n", latestRecord

      CALL readWarehouseRecord(warehouseFile, latestRecord, &S1)
      CALL printInventoryData(&S1)
      CALL closeWarehouseFile(warehouseFile)
 ENDFUNCTION
```

partsSupplier.h

```
IFNDEF partSupplier_Header
DEFINE partSupplier_Header

STRUCT partsSupplier
     char supplierID[10], supplierName[20], location[16],    partID[10], partName[20], linefeed1
ENDSTRUCT

void printSupplierData(partsSupplier* sup)
void partsSupplierCreation()

ENDIF
```

partsSupplier.c

```
INCLUDE "partsSupplier.h"
INCLUDE "fileHandling.h"

FUNCTION void printSupplierData(partsSupplier* sup)
     print "Supplier ID\t: %s\n", sup->supplierID
     print "Supplier Name\t: %s\n", sup->supplierName
     print "Location\t: %s\n", sup->location
     print "Part ID\t\t: %s\n", sup->partID
     print "Part Name\t: %s\n", sup->partName
ENDFUNCTION

FUNCTION void partsSupplierCreation()
     FILE* supplierFile
     partsSupplier S2
     DECLARE int latestRecord = 0

     print"2. Parts Supplier Creation\n- - - - -\n"

     memset(&S2, 0, sizeof(partsSupplier))
     supplierFile = CALL openSupplierFile()

     print "*Input existing Supplier ID if there is any\nEnter Supplier ID\t: "
     read S2.supplierID

     print "\nEnter Supplier Name\t: "
     read S2.supplierName

     print "\nEnter location\t\t: "
     read S2.location
     print "\nEnter Part ID\t\t: "
     read S2.partID

     print "\nEnter Part Name\t\t: "
     read S2.partName

     S2.linefeed1 = '\n'

     latestRecord = CALL addSupplierRecord(supplierFile, &S2)
     print "\nRecord successfully created!\n- - - - -\n"

     latestRecord = CALL getSupplierRecordCount(supplierFile)
     print "Current record count: %d\n", latestRecord

     CALL readSupplierRecord(supplierFile, latestRecord, &S2)
     CALL printSupplierData(&S2)

     CALL closeSupplierFile(supplierFile)
ENDFUNCTION
```

partsInventoryUpdate.h

```
IFNDEF partInventoryUpdate_Header
DEFINE partInventoryUpdate_Header

void partsInventoryUpdate()

ENDIF
```

partsInventoryUpdate.c

```
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"



FUNCTION int partsInventoryFind(FILE* warehouseFile, partsInventory *S1, char *partID)
     DECLARE int i = 0
     int totalRecord = CALL getWarehouseRecordCount(warehouseFile)
     memset(S1, 0, sizeof(partsInventory))
     FOR i < totalRecord
          CALL readWarehouseRecord(warehouseFile, i, S1)
          IF (strcmp(partID, S1->partID) == 0) THEN
            RETURN i
          ENDIF
        i = i + 1
     ENDFOR
      RETURN -1
ENDFUNCTION

FUNCTION void strAdd(char* strQuantity, int addValue)
     DECALRE int intQuantity
     intQuantity = atoi(strQuantity)
     intQuantity = intQuantity + addValue
     stringprint(strQuantity = intQuantity)
ENDFUNCTION


FUNCTION int strSubstract(char* strQuantity, int substractValue)
     DECLARE int intQuantity
     intQuantity = atoi(strQuantity)
     IF (intQuantity < substractValue) THEN
          RETURN -1
    ENDIF
    intQuantity = intQuantity - substractValue
    stringprint(strQuantity, "%d", intQuantity)
    RETURN 0
ENDFUNCTION
```

```
FUNCTION void partsInventoryUpdate()
     FILE* warehouseFile
     partsInventory S1
     DECLARE int quantityUpdate = 0, searchResult
     DECLARE char selectID[10], warehouseCode[5], updateOption[10]

     print "1. Parts Inventory Update\n- - - - -\n"
reinputwarehouse:
     memset(&S1, 0, sizeof(partsInventory))
     print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: "
     read warehouseCode
     print "1. Parts Inventory Update\n- - - - -\n"
reinputwarehouse:
     memset(&S1, 0, sizeof(partsInventory))
     print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: "
     read warehouseCode

     IF (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) THEN
          strcpy(warehouseCode, "WBZ");
     ELSE IF (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL"))
          strcpy(warehouseCode, "WSL")
     ELSE IF (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR"))
          strcpy(warehouseCode, "WAR")
     ELSE
          printf "\nInvalid warehouse code. Please try again\n"
          goto reinputwarehouse;
     ENDIF

     warehouseFile = CALL openWarehouseFile(warehouseCode)
     print "\nInput part ID to update: "
     read selectID

     print "Search Result\n- - - - - \n"

     searchResult = CALL partsInventoryFind(warehouseFile, &S1, selectID)
     CALL printInventoryData(&S1)

     print "\n1. Add Quantity\n2. Substract Quantity\nSelect Update Option: "
     read updateOption

reinputquantity:
     print "\nInput Quantity: "
     read quantityUpdate

     IF (0 == strcmp(updateOption, "1") || 0 == strcmp(updateOption, "Add")) THEN
          CALL strAdd(S1.quantity, quantityUpdate)
     ELSE IF (0 == strcmp(updateOption, "2") || 0 == strcmp(updateOption, "Substract"))

          IF (-1 == CALL strSubstract(S1.quantity, quantityUpdate)) THEN
               print "\nInsufficient quantity to be substracted. Try input the value again"
               goto reinputquantity
          ENDIF
     ENDIF

     CALL updateWarehouseRecord(warehouseFile, searchResult, &S1)
     print "\nUpdated info\n- - - - -\n"
     CALL printInventoryData(&S1)
     CALL closeWarehouseFile(warehouseFile)

ENDFUNCTION
```

partsInventoryTracking.h

```
IFNDEF partsInventoryTracking_header
DEFINE partsInventoryTracking_header

void partsInventoryTracking()

ENDIF
```

partsInventoryTracking.c

```
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"

FUNCTION int sortPartsAsc(char *warehouseFileName, partsInventory *S1List, int S1ListSize)
    FILE* warehouseFile
    partsInventory S1
    DECLARE int totalRecord, arraySize = 0, S1ListCount = 0, j
    DECLARE int i = 0

    warehouseFile = CALL openWarehouseFile(warehouseFileName)
    totalRecord = CALL getWarehouseRecordCount(warehouseFile)

    FOR i < totalRecord
        CALL readWarehouseRecord(warehouseFile, i, &S1)

        IF (S1ListCount == 0) THEN
            memcpy(S1List + S1ListCount, &S1, 64)
        ELSE
            j = S1ListCount
            FOR j > 0
                IF (0 < strcmp((S1List+ j - 1)->partID, S1.partID)) THEN
                    memcpy(S1List + j, S1List + j - 1, sizeof(partsInventory))
                ELSE
                    memcpy(S1List + j, &S1, sizeof(partsInventory))
                    BREAK
                ENDIF
                j = j - 1
            ENDFOR
            IF (j == 0) THEN
                memcpy(S1List, &S1, sizeof(partsInventory))
            ENDIF
        ENDIF
        i = i +1
        S1ListCount += 1
    ENDFOR
    CALL closeWarehouseFile(warehouseFile)

    IF (S1ListCount > 0) THEN
        return totalRecord
    ENDIF
    RETURN 0
ENDFUNCTION
```

```
FUNCTION int quantityLessThan10(char* warehouseFileName, partsInventory* S1List, int S1ListSize)
    FILE* warehouseFile
    partsInventory S1
   DECLARE int totalRecord, arraySize = 0, S1ListCount = 0, j
   DECLARE int i = 0
    memset(S1List, 0, S1ListSize * sizeof(partsInventory))

    warehouseFile = CALL openWarehouseFile(warehouseFileName)

    totalRecord = CALL getWarehouseRecordCount(warehouseFile)

    FOR (i < totalRecord)
        CALL readWarehouseRecord(warehouseFile, i, &S1)

        IF (atoi(S1.quantity) < 10) THEN
            memcpy(S1List + S1ListCount, &S1, sizeof(partsInventory));
            S1ListCount++;
       ENDIF
     i = i + 1
    ENDFOR

    CALL closeWarehouseFile(warehouseFile);

    IF (S1ListCount > 0) THEN
        RETUNRN S1ListCount
    ENDIF
    RETURN 0
ENDFUNCTION


FUNCTION void partsInventoryTracking()
    partsInventory S1List[100]
    DECLARE int recordCount, i = 0, j =0
    DECLARE char trackingOption[5], warehouseCode[5];
    DECLARE char warehouseFile[3][5] = {"WBZ", "WSL", "WAR"}

    print "Parts Inventory Tracking\n- - - - -\n"
reinputOption:
    print "\n1. Sort all parts in warehouses\n2. Show stock less than 10 quantity\n\nSelect function, input (1) or (2): "
    read trackingOption

    IF (0 == strcmp(trackingOption, "1")) THEN
        print "\n1. Sort all parts\n"
        print "Sorted Warehouse Data"
        print "\n- - - - -\n"
        FOR (i < 3)
            memset(&S1List, 0, 100 * sizeof(partsInventory));
              recordCount = CALL sortPartsAsc(warehouseFile[i], &S1List[0], 100)

              print "\n\nWarehouse Code: %s\n", warehouseFile[i]
              print "\nRecord count = %d\n", recordCount
              print "Part ID\t| Section Code\t| Part Name\t\t| Quantity\t\n"
              FOR (int j = 0; j < recordCount; j++)
                  print "%s\t| ", S1List[j].partID
                  print "%s\t\t| ", S1List[j].sectionCode
                  print "%s\t\t| ", S1List[j].partName
                  print "%s\t\n", S1List[j].quantity
              ENDFOR
              i = i + 1
         ENDFOR
     ELSE IF (0 == strcmp(trackingOption, "2"))
         print "\n\n2. Show stock less than 10 quantity\n- - - - -\n"
reinputwarehouse:
         print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: "
         read warehouseCode

         IF (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) THEN
             strcpy(warehouseCode, "WBZ")
         ELSE IF (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL"))
             strcpy(warehouseCode, "WSL")
         ELSE IF (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR"))
             strcpy(warehouseCode, "WAR")
```

```
        ELSE
            print "\n\nInvalid warehouse code. Please try again\n"
            goto reinputwarehouse
        ENDIF

        memset(&S1List, 0, 100 * sizeof(partsInventory))
        recordCount = quantityLessThan10(warehouseCode, &S1List[0], 100)
        print "Stock Quantity that has less then 10 units"
        print "\n- - - - -\n"
        print "\nWarehouse Code: %s\n", warehouseCode
        print "Record Count = %d\n\n", recordCount
        print "Part ID\t| Section Code\t| Part Name\t\t| Quantity\t\n"
        FOR (j < recordCount)
             print "%s\t| ", S1List[j].partID
             print "%s\t\t| ", S1List[j].sectionCode
             print "%s\t\t| ", S1List[j].partName
             print "%s\t\n", S1List[j].quantity
             j = j + 1
        ENDFOR
    ELSE
        print "\nInput invalid, try again with (1) or (2)\n"
        GOTO reinputOption
    ENDIF
ENDFUNCTION
```

searchingFunctionalities.h

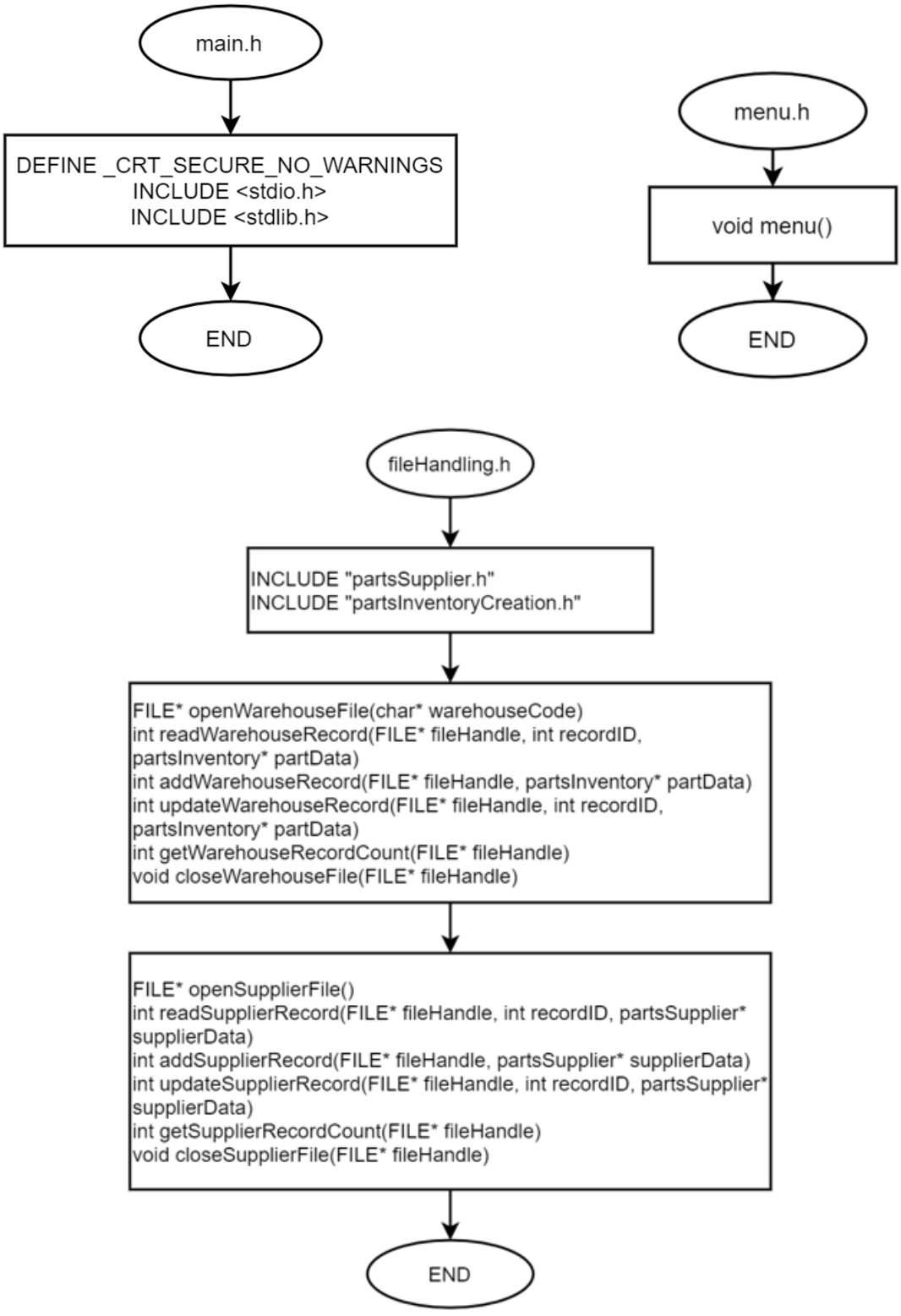
```
IFNDEF searchingFunctionalities_Header
DEFINE searchingFunctionalities_Header

void searchingFunctionalities()

ENDIF
```

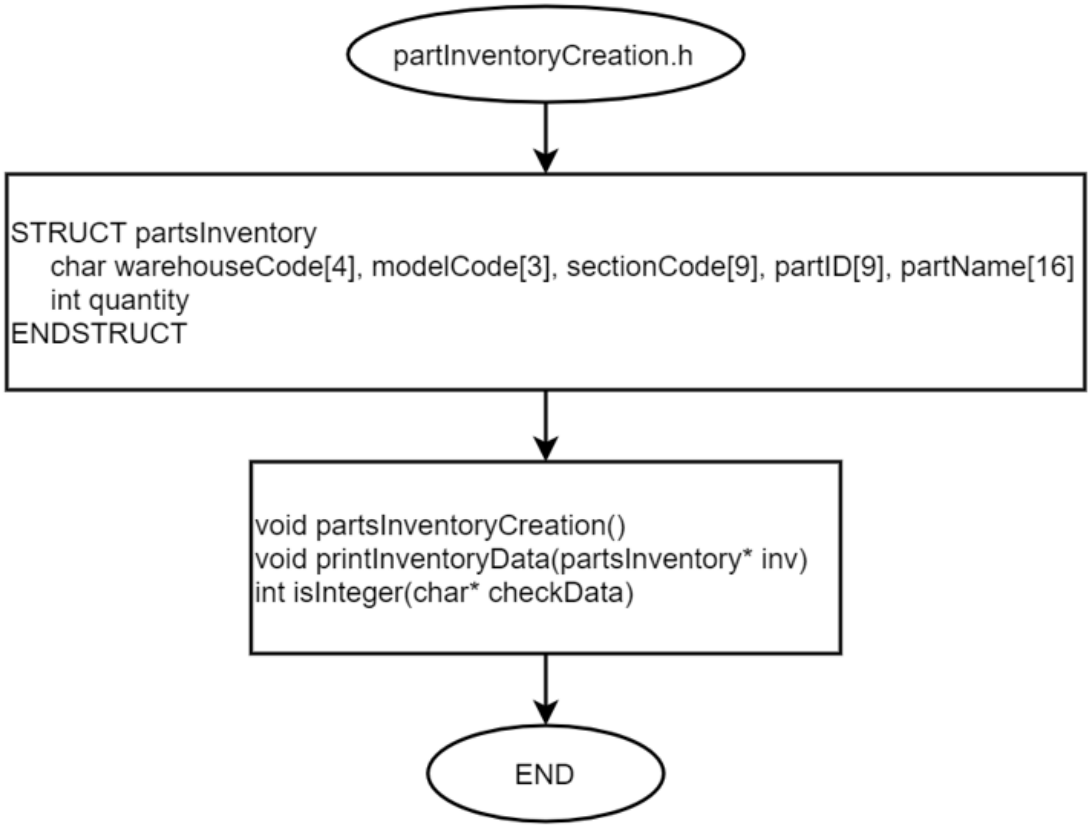
searchingFunctionalities.c

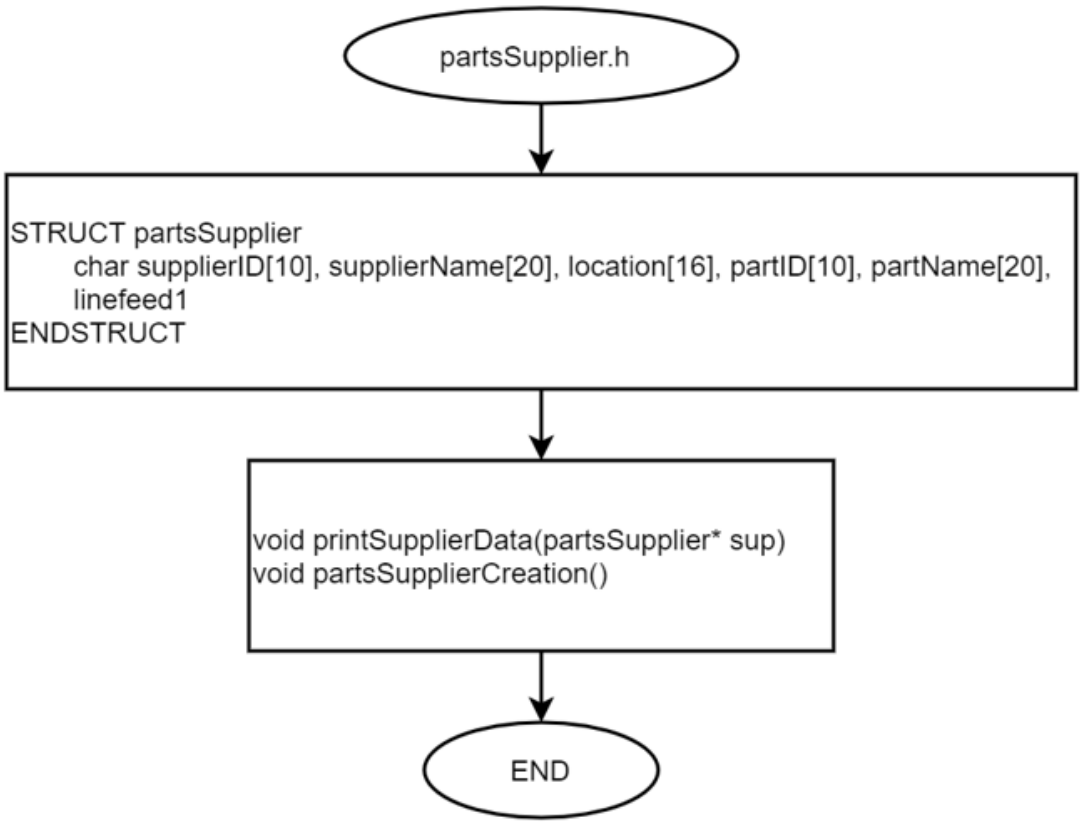
```
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"
INCLUDE "partsSupplier.h"

FUNCTION int partsInventorySearch(FILE* warehouseFile, partsInventory* S1, char* partID)
     DECLARE int totalRecord, i = 0
     totalRecord = getWarehouseRecordCount(warehouseFile)

     memset(S1, 0, sizeof(partsInventory))
     FOR (i < totalRecord)
          readWarehouseRecord(warehouseFile, i, S1)
          IF (strcmp(partID, S1->partID) == 0) THEN
               RETURN i
          ENDIF
          i = i + 1
     ENDFOR
     RETURN -1
ENDFUNCTION


FUNCTION int partsSupplierSearch(FILE* supplierFile, partsSupplier* S2, char* partID)
     DECLARE int totalRecord, i = 0
     totalRecord = getSupplierRecordCount(supplierFile)

     memset(S2, 0, sizeof(partsSupplier))
     FOR (i < totalRecord; i++)
          readSupplierRecord(supplierFile, i, S2);
          IF (strcmp(partID, S2->partID) == 0) THEN
               return i;
          ENDIF
          i = i + 1
     ENDFOR
     RETURN -1
ENDFUNCTION


FUNCTION void searchingFunctionalities()
     FILE* warehouseFile;
     FILE* supplierFile;
     partsInventory S1;
     partsSupplier S2;
     DECLARE char selectID[10], warehouseCode[5], searchOption[15], toMenu[3]
     DECLARE int searchResult

     print "5. Searching Functionalities\n- - - - -\n"

selectSearchFunction:
     print "1. Part Record\n2. Supplier Details\nSelect Search Function: "
     read searchOption
```

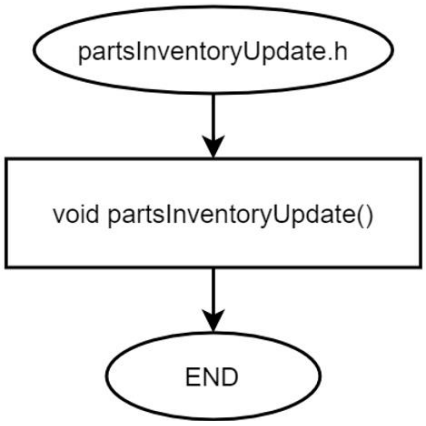
```
     IF (0 == strcmp(searchOption, "1") || 0 == strcmp(searchOption, "Part Record")) THEN
 reinputwarehouse:
          memset(&S1, 0, sizeof(partsInventory))
          print "\nWarehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: ")
          read warehouseCode

          IF (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) THEN
               strcpy(warehouseCode, "WBZ")
          ELSE IF (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL"))
               strcpy(warehouseCode, "WSL")
          ELSE IF (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR"))
               strcpy(warehouseCode, "WAR")
          ELSE
               print "\nInvalid warehouse code. Please try again\n"
               GOTO reinputwarehouse
          ENDIF

          warehouseFile = CALL openWarehouseFile(warehouseCode)

          print "\nInput Part ID to search: "
          read selectID

          print "\n\nSearch Result\n- - - - - \n"

          searchResult = CALL partsInventorySearch(warehouseFile, &S1, selectID)
          CALL printInventoryData(&S1)
          CALL closeWarehouseFile(warehouseFile)
     ELSE IF (0 == strcmp(searchOption, "2") || 0 == strcmp(searchOption, "Supplier Details"))
          memset(&S2, 0, sizeof(partsSupplier))

          supplierFile = CALL openSupplierFile()

          print "\nInput Part ID to search: "
          read selectID

          print "\n\nSearch Result\n- - - - - \n"

          CALL partsSupplierSearch(supplierFile, &S2, selectID)
          CALL printSupplierData(&S2)

          CALL closeSupplierFile(supplierFile)
     ELSE
          print "\nInput invalid, try again with (1) or (2)\n"
          GOTO selectSearchFunction
     ENDIF
ENDFUNCTION
```

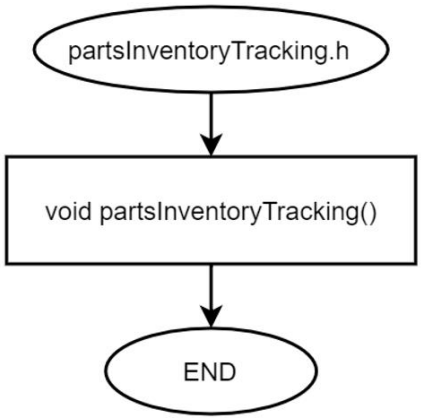
2.2 Flowchart

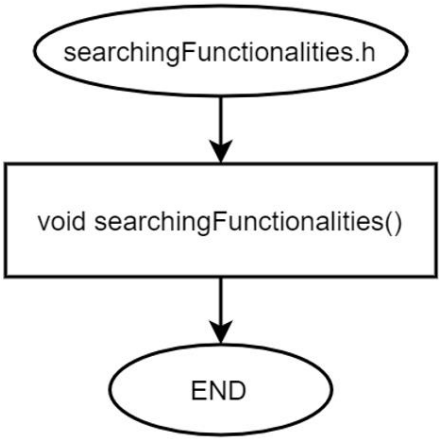


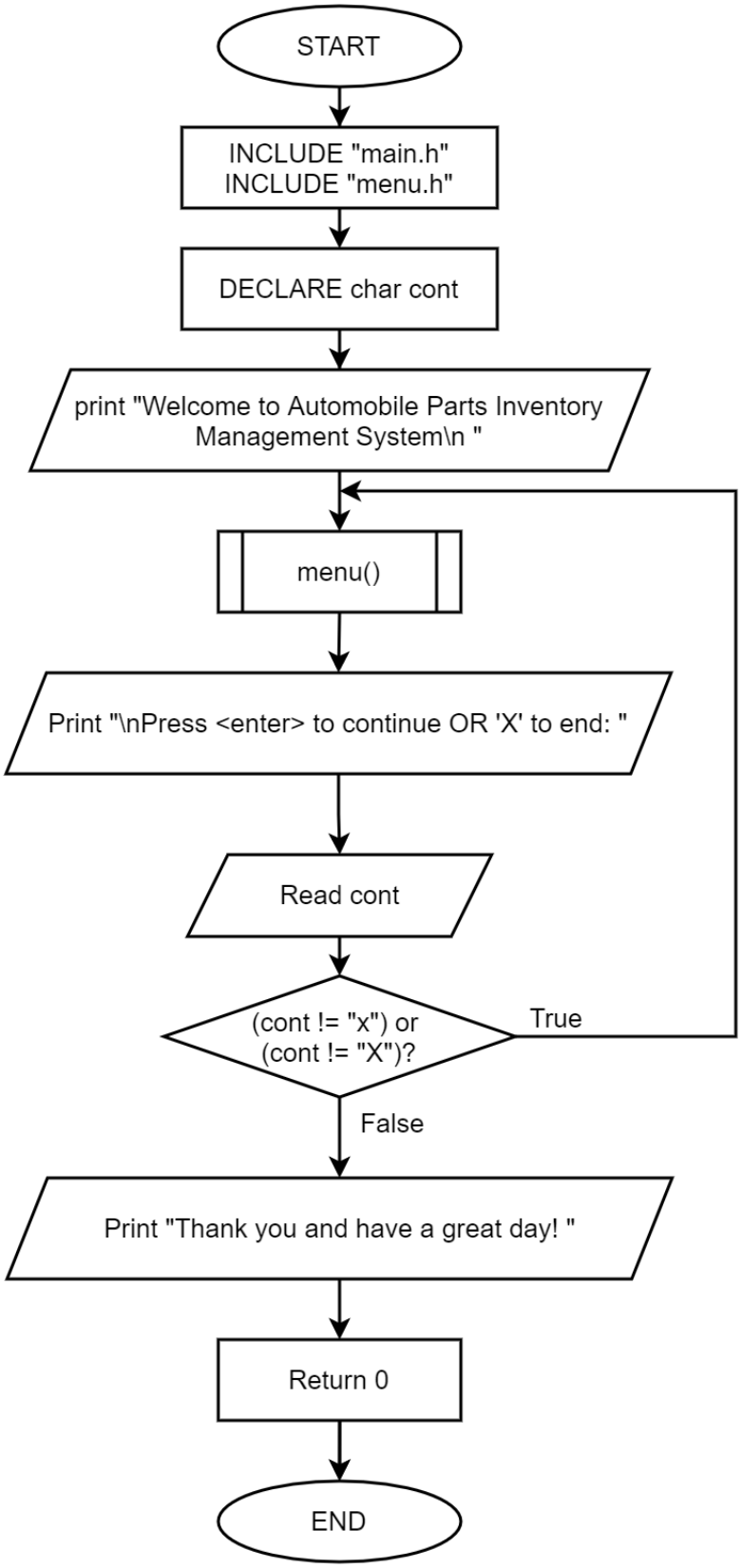

partInventoryCreation.h
STRUCT partsInventory
char warehouseCode[4], modelCode[3], sectionCode[9], partID[9], partName[16]
int quantity
ENDSTRUCT
void partsInventoryCreation()
void printInventoryData(partsInventory* inv)
int isInteger(char* checkData)
END

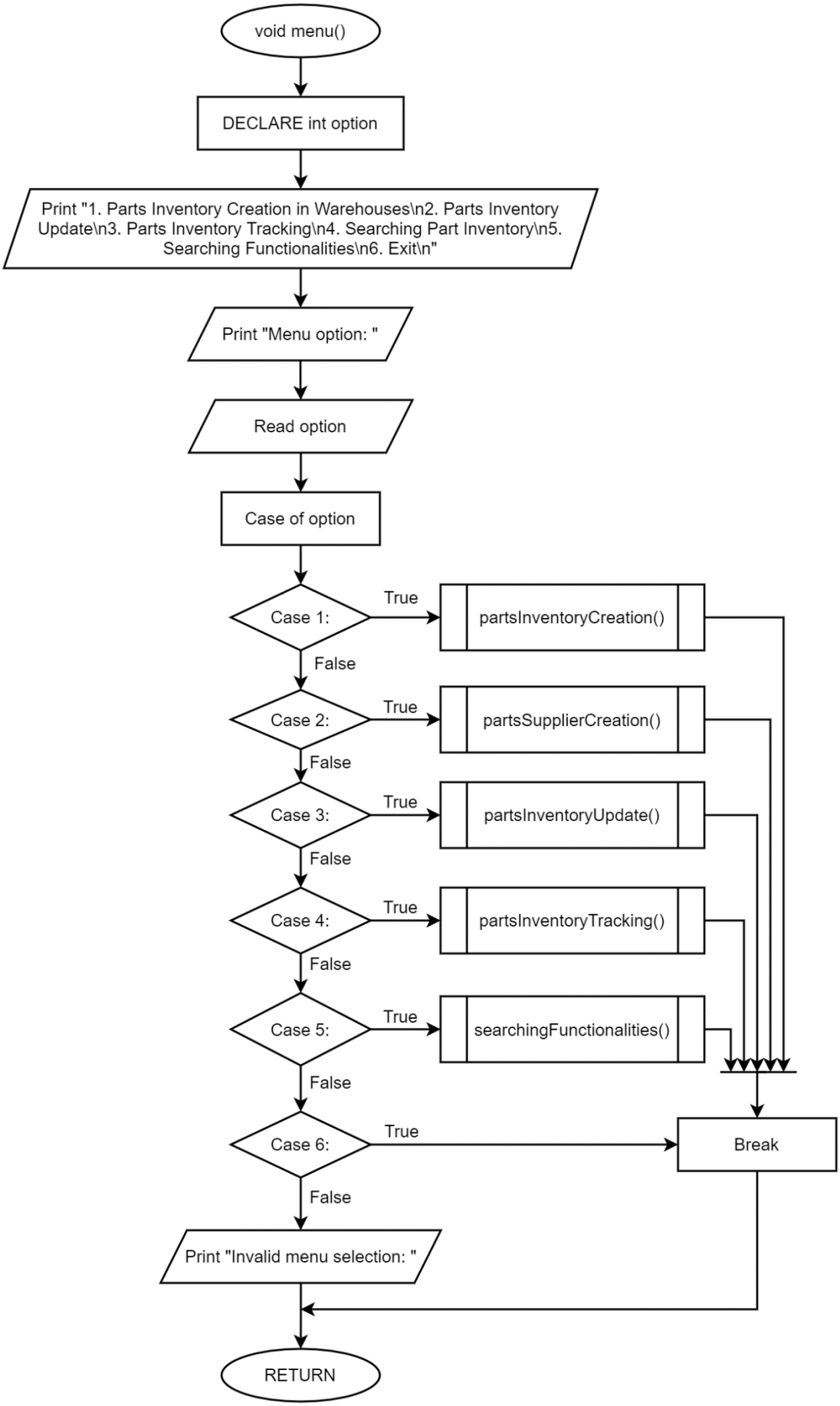

partsSupplier.h
STRUCT partsSupplier
char supplierID[10], supplierName[20], location[16], partID[10], partName[20],
linefeed1
ENDSTRUCT
void printSupplierData(partsSupplier* sup)
void partsSupplierCreation()
END

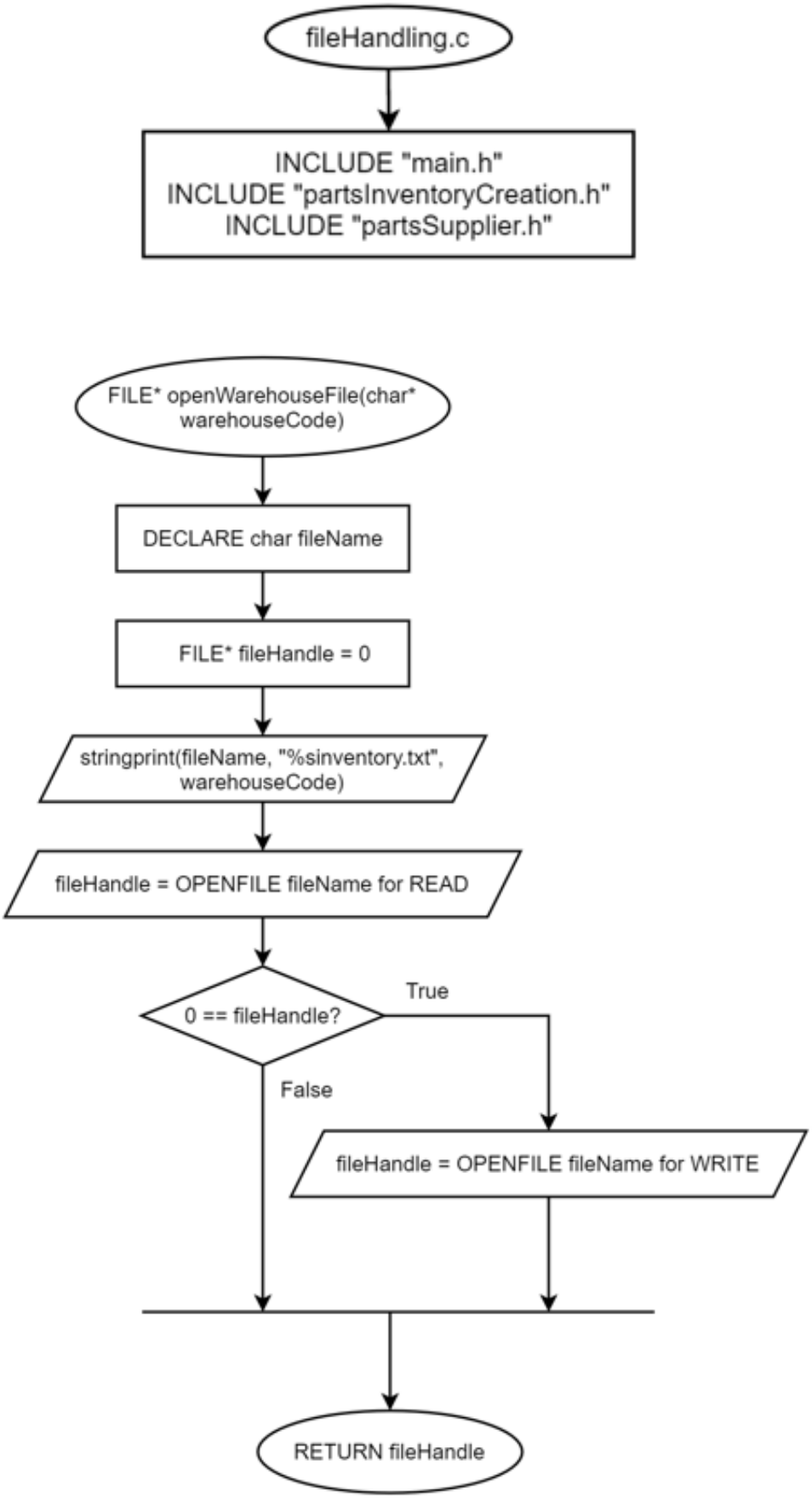

partsInventoryUpdate.h
void partsInventoryUpdate()
END

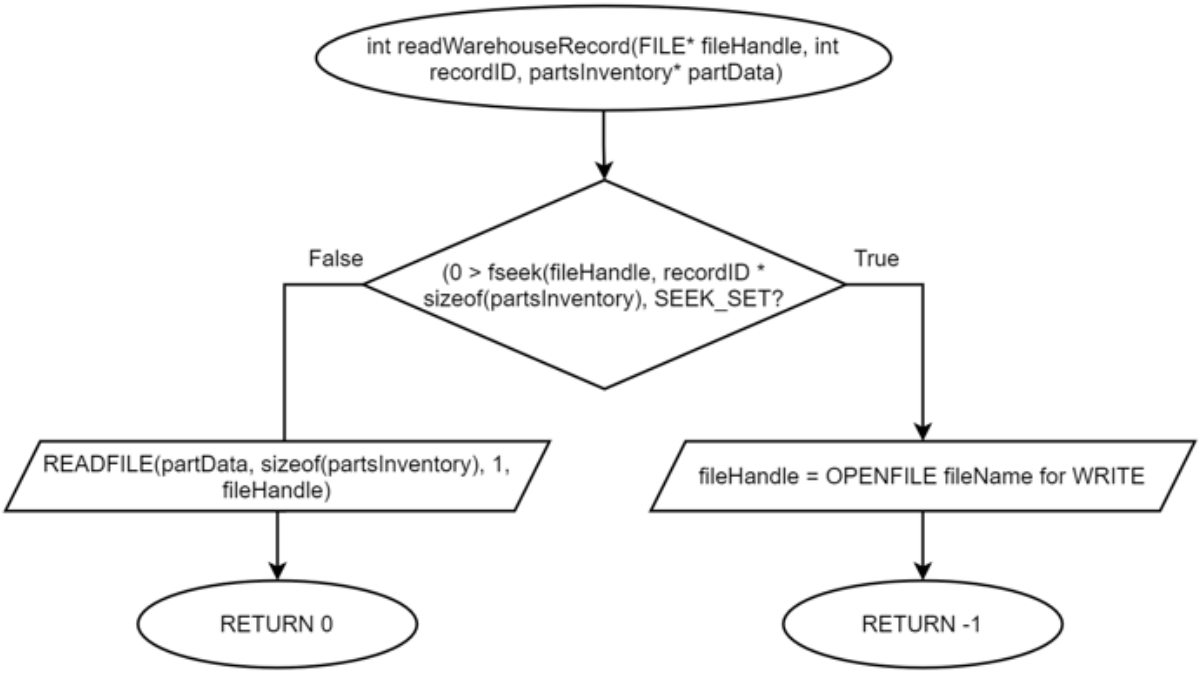

partsInventoryTracking.h
void partsInventoryTracking()
END

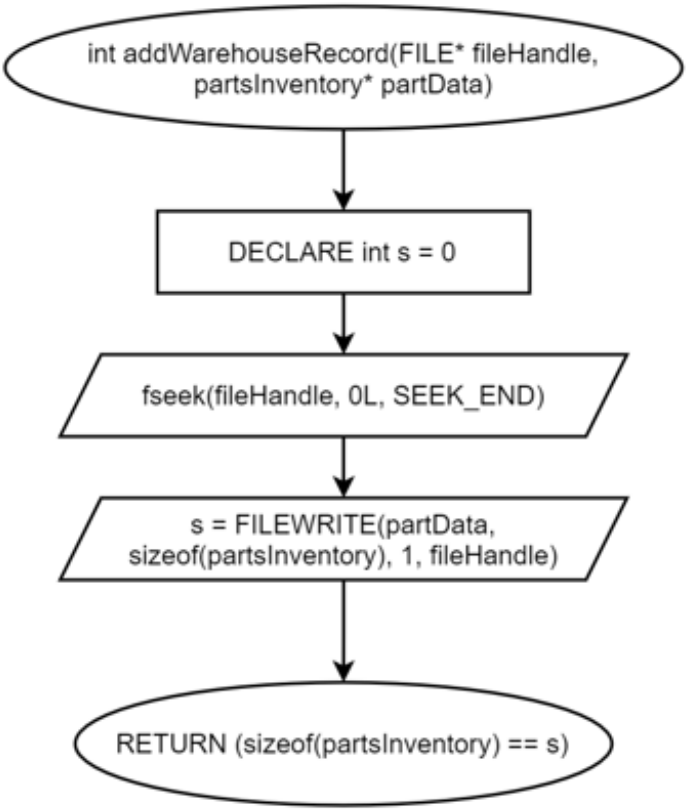

searchingFunctionalities.h
void searchingFunctionalities()
END

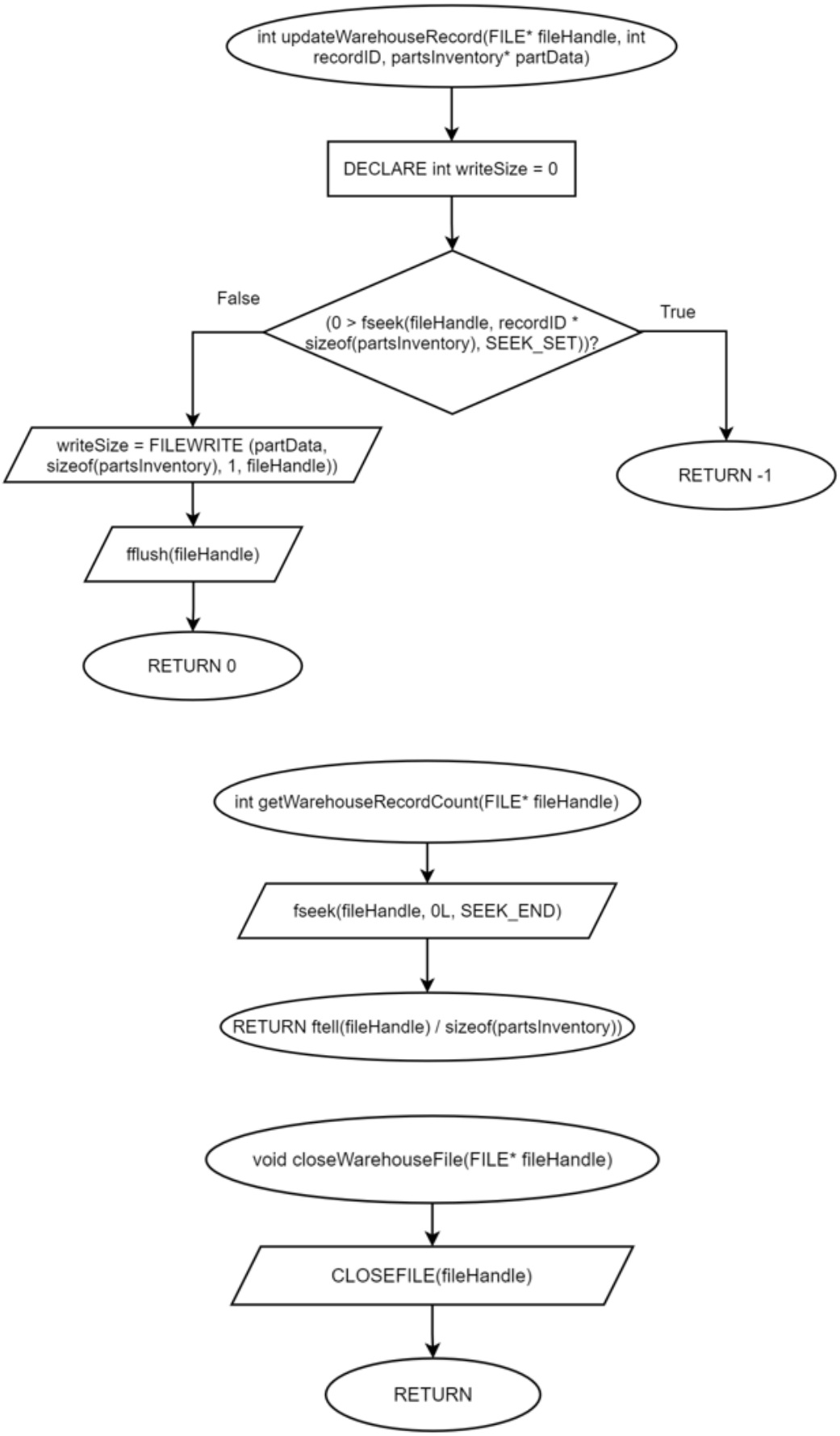

START
INCLUDE "main.h"
INCLUDE "menu.h"
DECLARE char cont
print "Welcome to Automobile Parts Inventory Management System\n "
menu()
Print "\nPress <enter> to continue OR 'X' to end: "
Read cont
(cont != "x") or (cont != "X")?
True
False
Print "Thank you and have a great day! "
Return 0
END

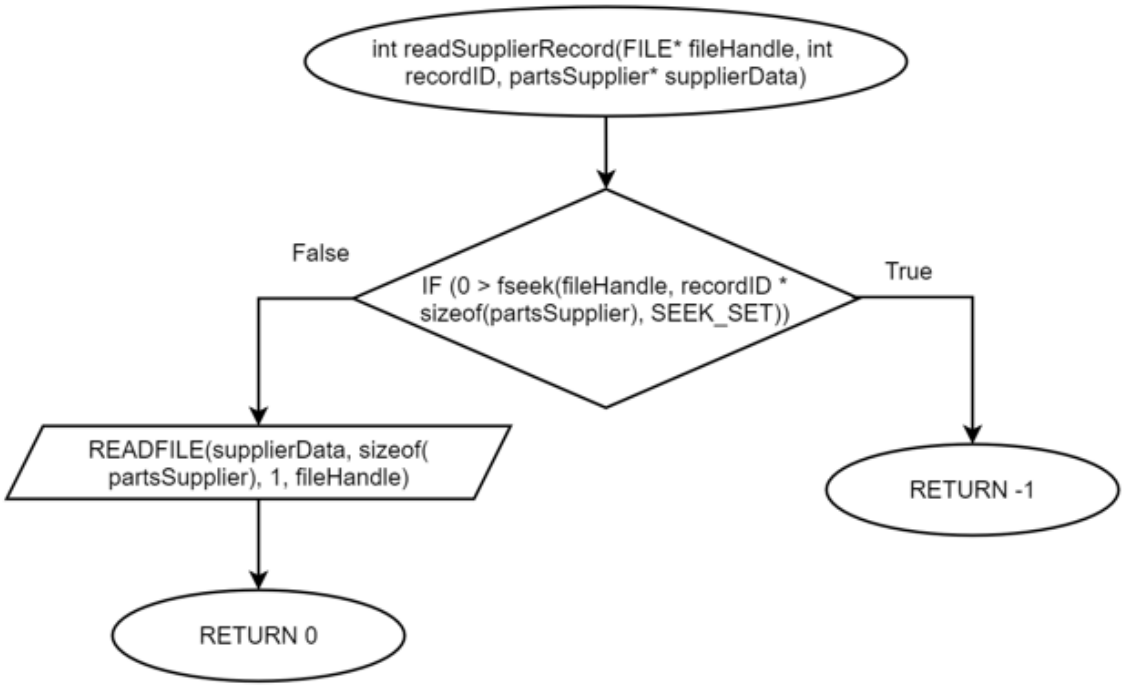

void menu()
DECLARE int option
Print "1. Parts Inventory Creation in Warehouses\n2. Parts Inventory Update\n3. Parts Inventory Tracking\n4. Searching Part Inventory\n5. Searching Functionalities\n6. Exit\n"
Print "Menu option: "
Read option
Case of option
Case 1:
True
partsInventoryCreation()
False
Case 2:
True
partsSupplierCreation()
False
Case 3:
True
partsInventoryUpdate()
False
Case 4:
True
partsInventoryTracking()
False
Case 5:
True
searchingFunctionalities()
False
Case 6:
True
Break
False
Print "Invalid menu selection: "
RETURN

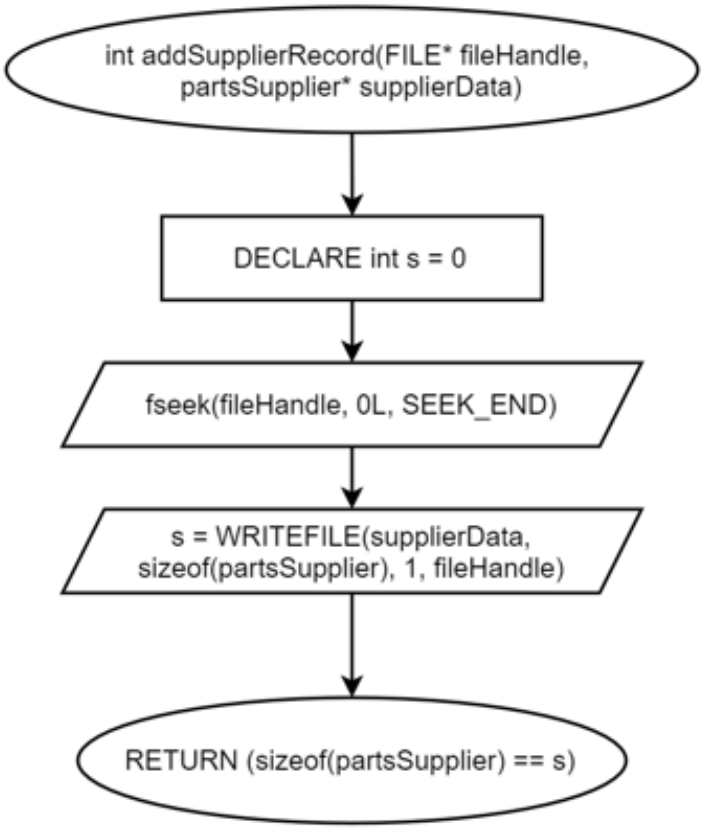

fileHandling.c
INCLUDE "main.h"
INCLUDE "partsInventoryCreation.h"
INCLUDE "partsSupplier.h"
FILE* openWarehouseFile(char* warehouseCode)
DECLARE char fileName
FILE* fileHandle = 0
stringprint(fileName, "%sinventory.txt", warehouseCode)
fileHandle = OPENFILE fileName for READ
0 == fileHandle?
True
False
fileHandle = OPENFILE fileName for WRITE
RETURN fileHandle

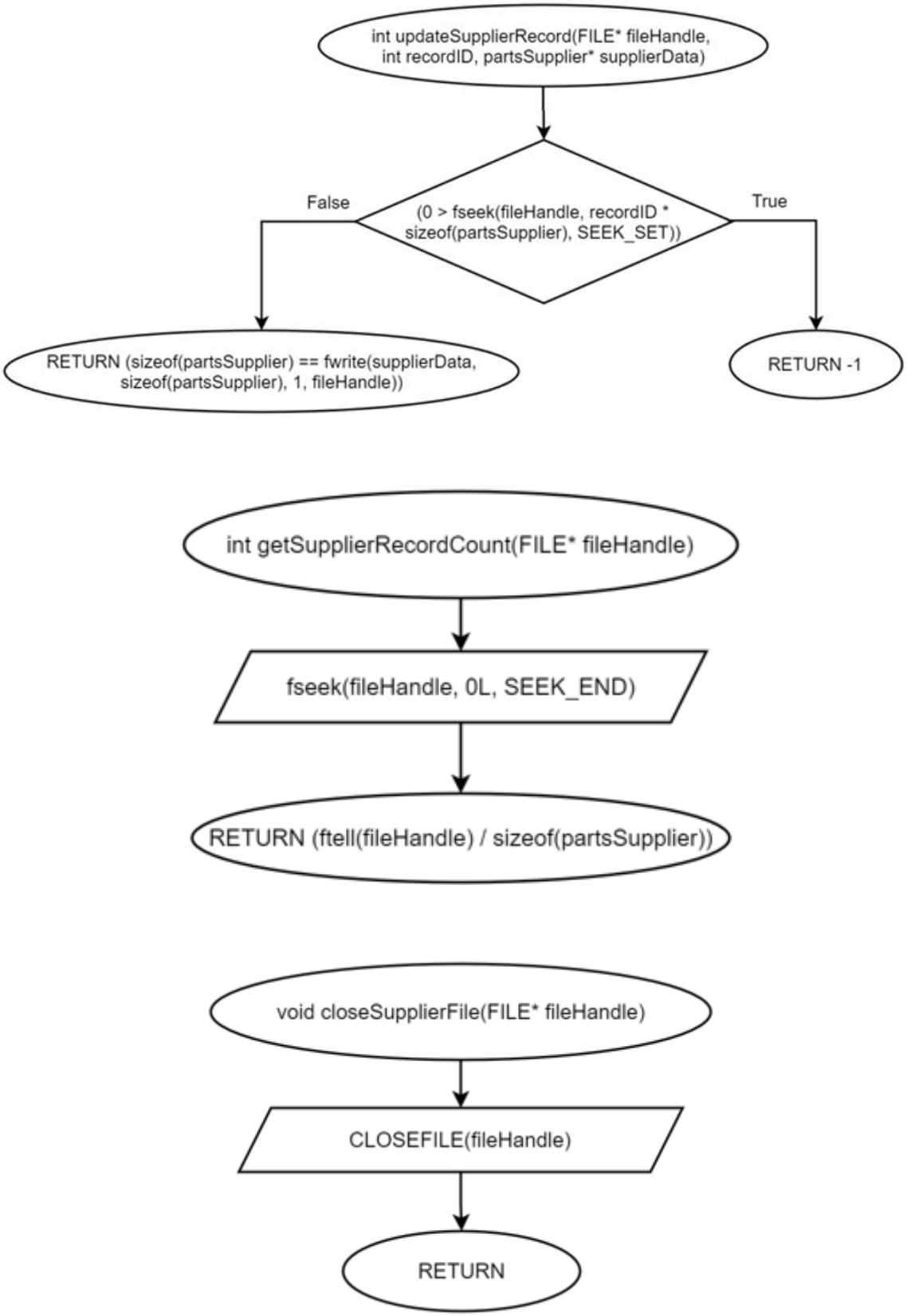

int readWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData)
(0 > fseek(fileHandle, recordID * sizeof(partsInventory), SEEK_SET?
False
True
READFILE(partData, sizeof(partsInventory), 1, fileHandle)
fileHandle = OPENFILE fileName for WRITE
RETURN 0
RETURN -1

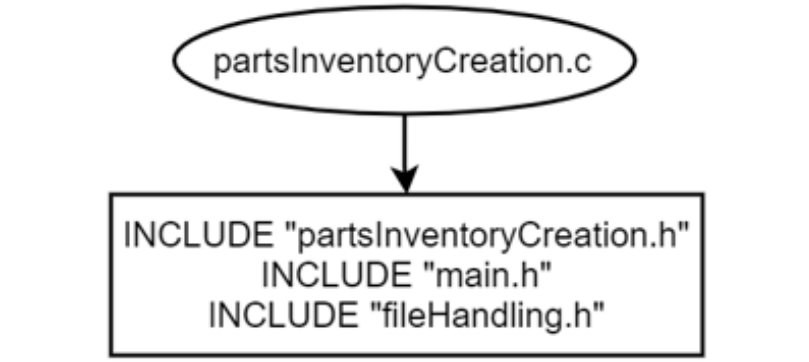

int addWarehouseRecord(FILE* fileHandle, partsInventory* partData)
DECLARE int s = 0
fseek(fileHandle, 0L, SEEK_END)
s = FILEWRITE(partData, sizeof(partsInventory), 1, fileHandle)
RETURN (sizeof(partsInventory) == s)

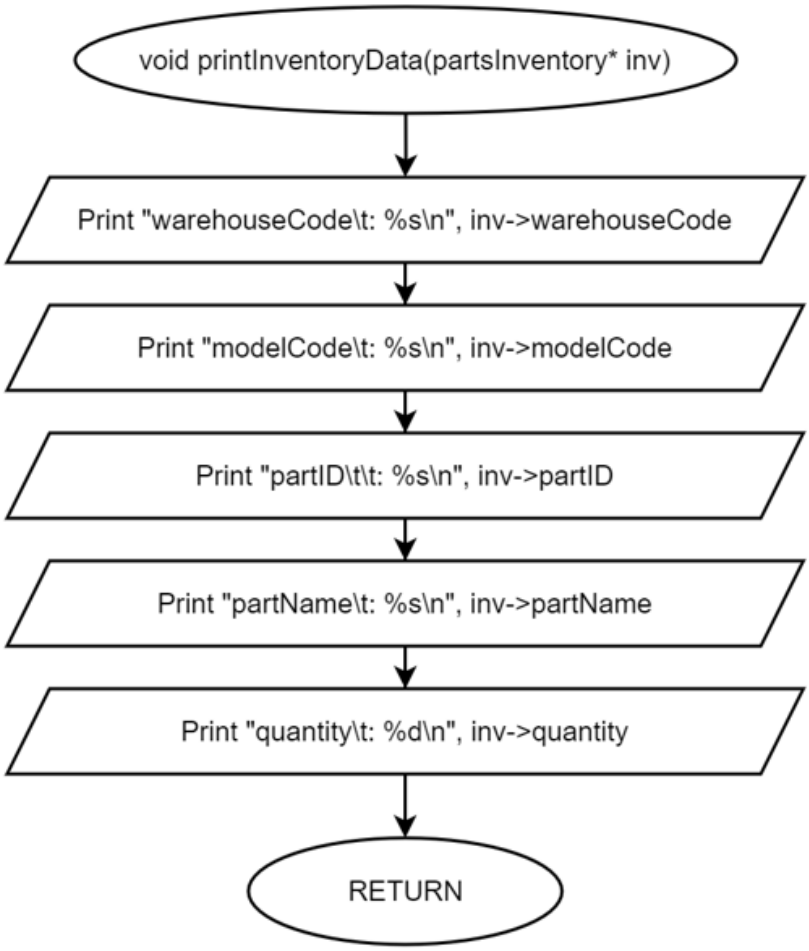

int updateWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData)
DECLARE int writeSize = 0
False
(0 > fseek(fileHandle, recordID * sizeof(partsInventory), SEEK_SET))?
True
writeSize = FILEWRITE (partData, sizeof(partsInventory), 1, fileHandle))
RETURN -1
fflush(fileHandle)
RETURN 0
int getWarehouseRecordCount(FILE* fileHandle)
fseek(fileHandle, 0L, SEEK_END)
RETURN ftell(fileHandle) / sizeof(partsInventory))
void closeWarehouseFile(FILE* fileHandle)
CLOSEFILE(fileHandle)
RETURN

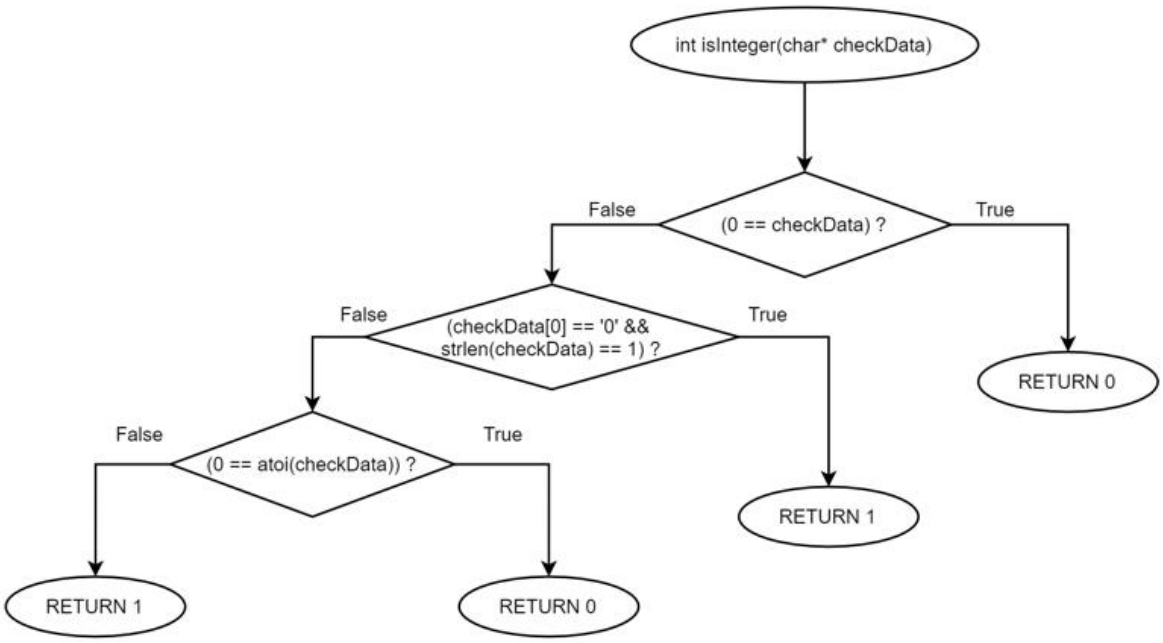

int readSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData)
IF (0 > fseek(fileHandle, recordID * sizeof(partsSupplier), SEEK_SET))
False
True
READFILE(supplierData, sizeof(partsSupplier), 1, fileHandle)
RETURN -1
RETURN 0

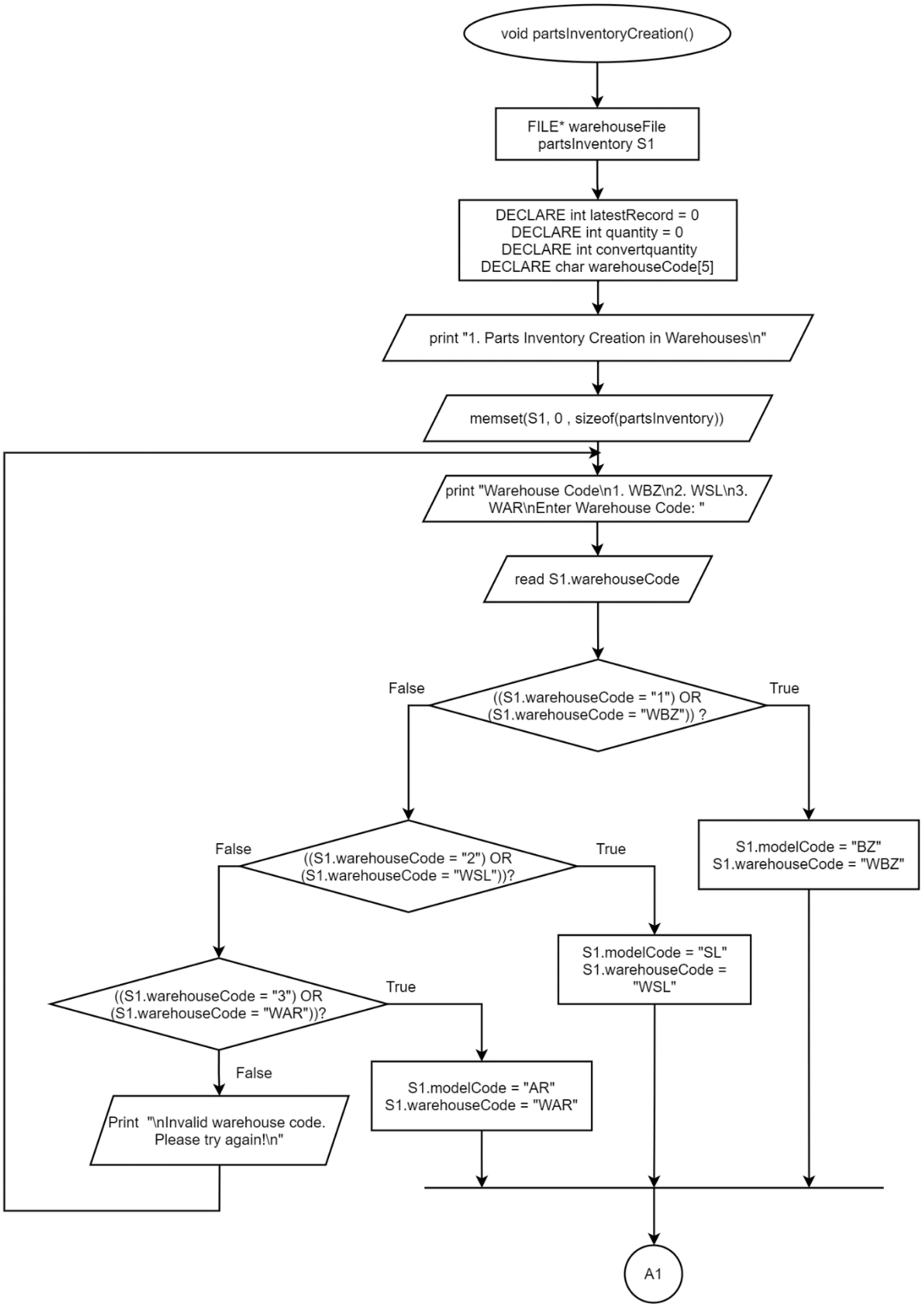

int addSupplierRecord(FILE* fileHandle, partsSupplier* supplierData)
DECLARE int s = 0
fseek(fileHandle, 0L, SEEK_END)
s = WRITEFILE(supplierData, sizeof(partsSupplier), 1, fileHandle)
RETURN (sizeof(partsSupplier) == s)

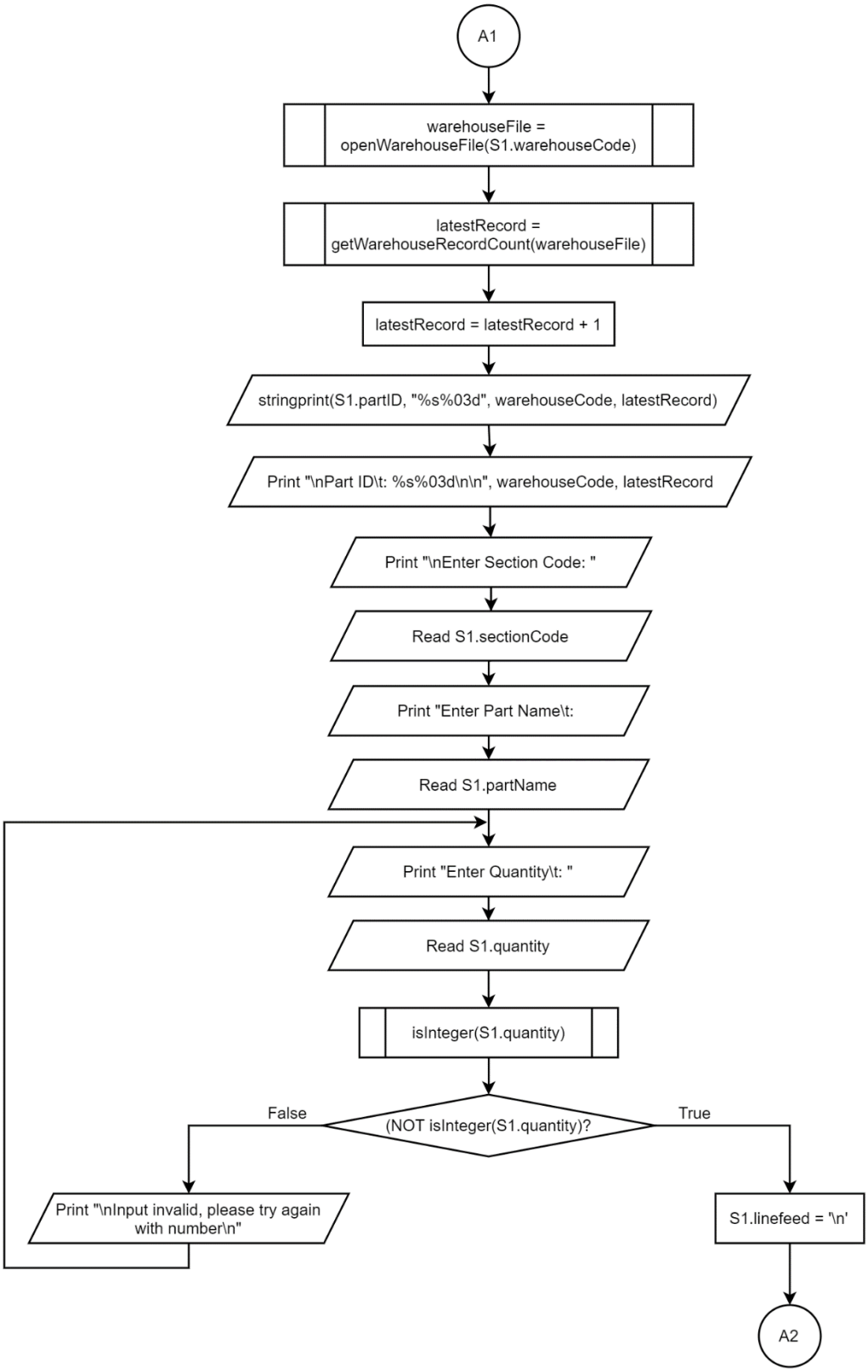

int updateSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData)
(0 > fseek(fileHandle, recordID * sizeof(partsSupplier), SEEK_SET))
False
True
RETURN (sizeof(partsSupplier) == fwrite(supplierData, sizeof(partsSupplier), 1, fileHandle))
RETURN -1
int getSupplierRecordCount(FILE* fileHandle)
fseek(fileHandle, 0L, SEEK_END)
RETURN (ftell(fileHandle) / sizeof(partsSupplier))
void closeSupplierFile(FILE* fileHandle)
CLOSEFILE(fileHandle)
RETURN

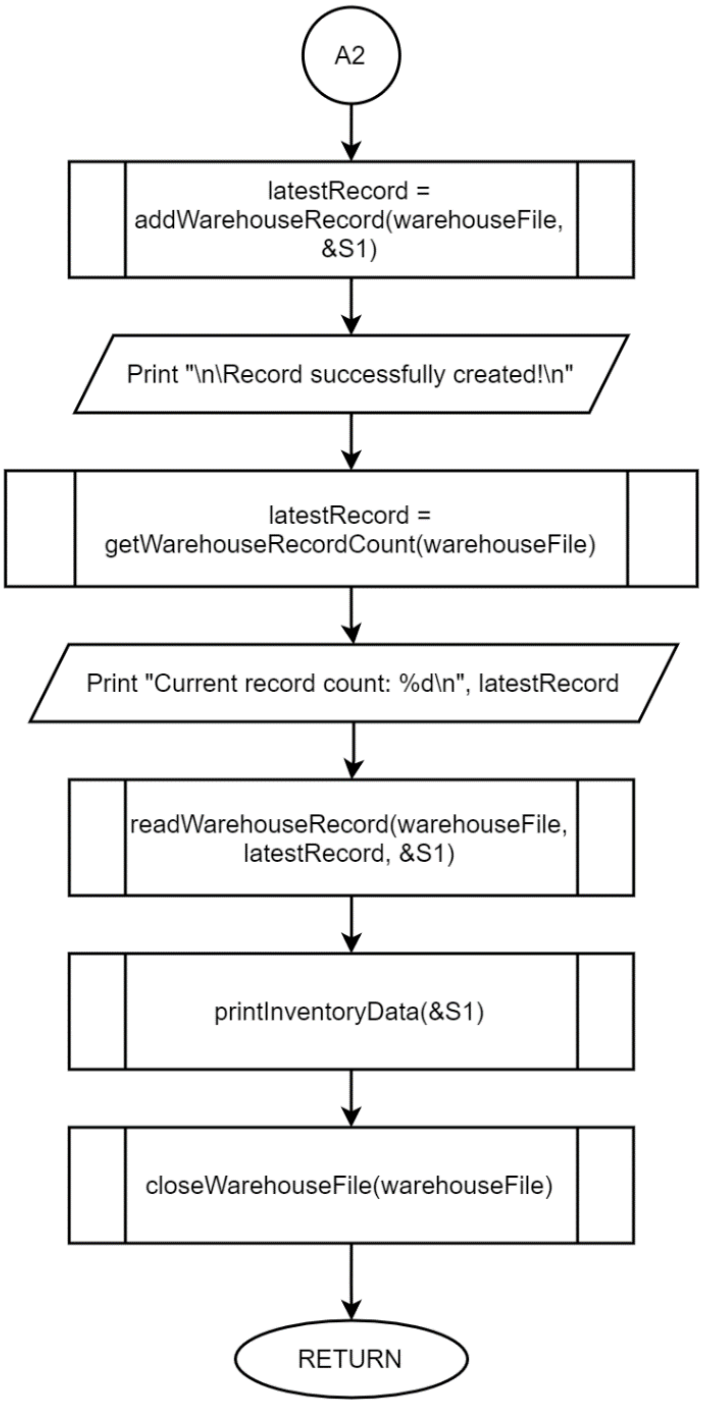

partsInventoryCreation.c
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"

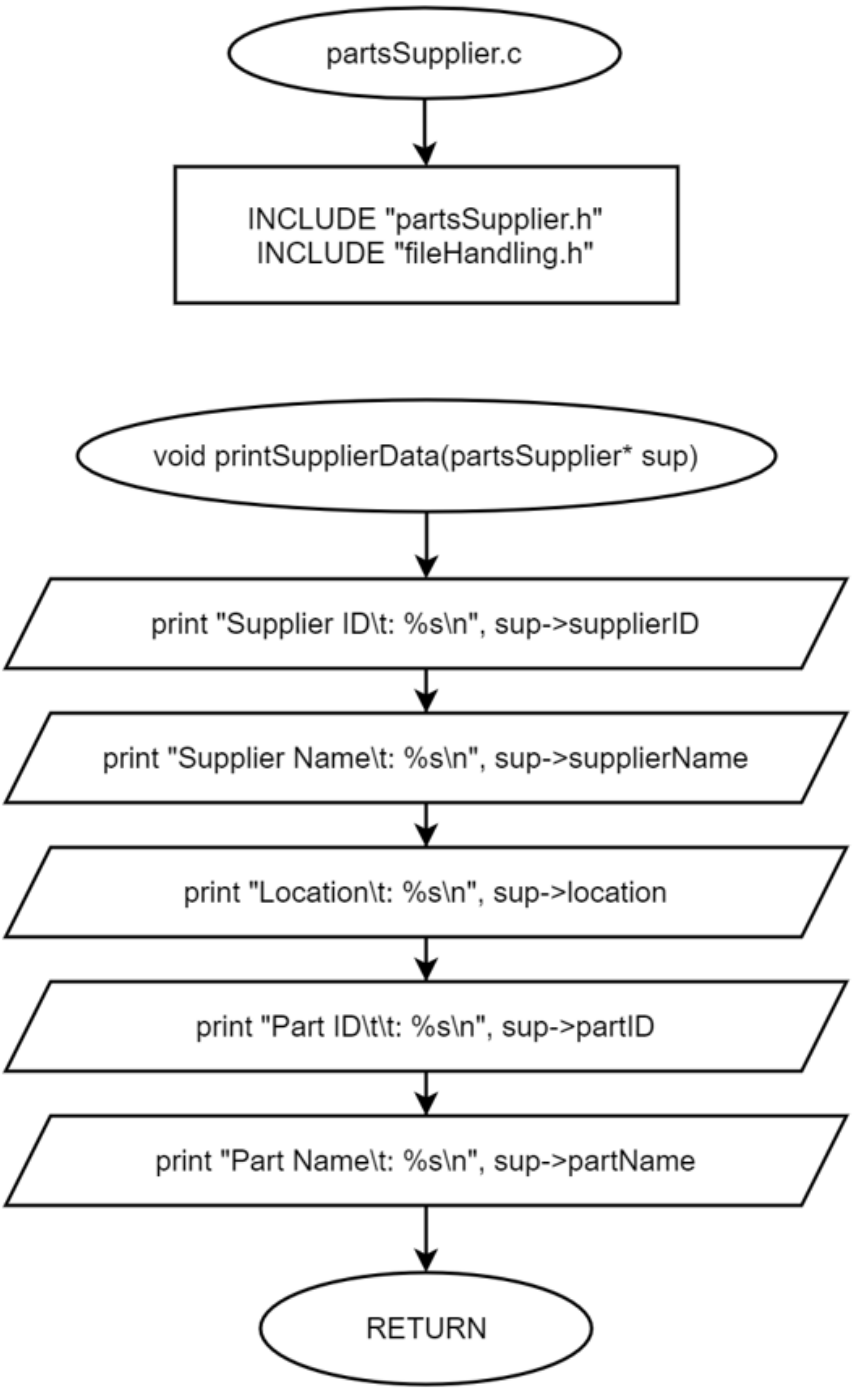

void printInventoryData(partsInventory* inv)
Print "warehouseCode\t: %s\n", inv->warehouseCode
Print "modelCode\t: %s\n", inv->modelCode
Print "partID\t\t: %s\n", inv->partID
Print "partName\t: %s\n", inv->partName
Print "quantity\t: %d\n", inv->quantity
RETURN

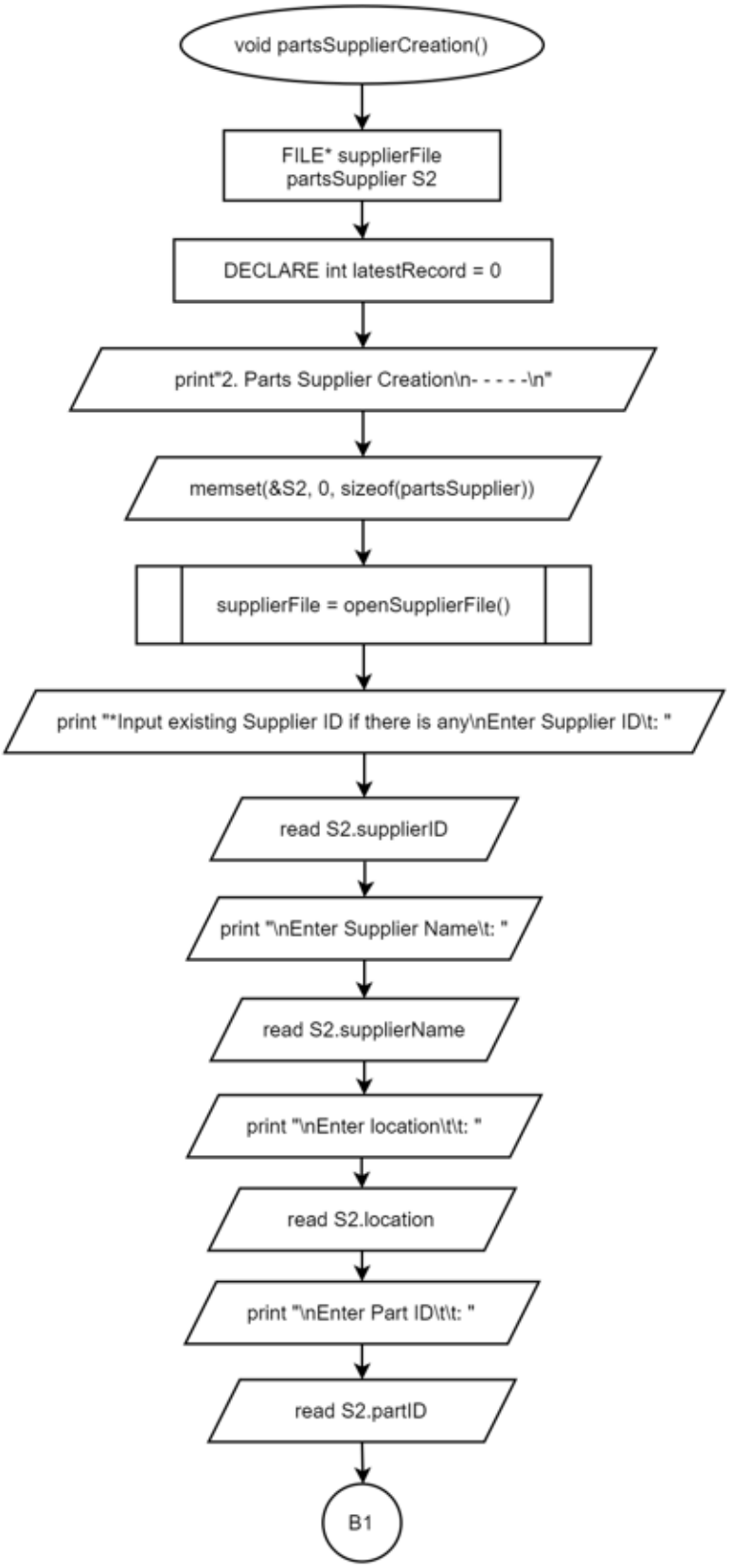

int isInteger(char* checkData)
(0 == checkData) ?
False
True
RETURN 0
(checkData[0] == '0' &&
strlen(checkData) == 1) ?
False
True
RETURN 1
(0 == atoi(checkData)) ?
False
True
RETURN 1
RETURN 0

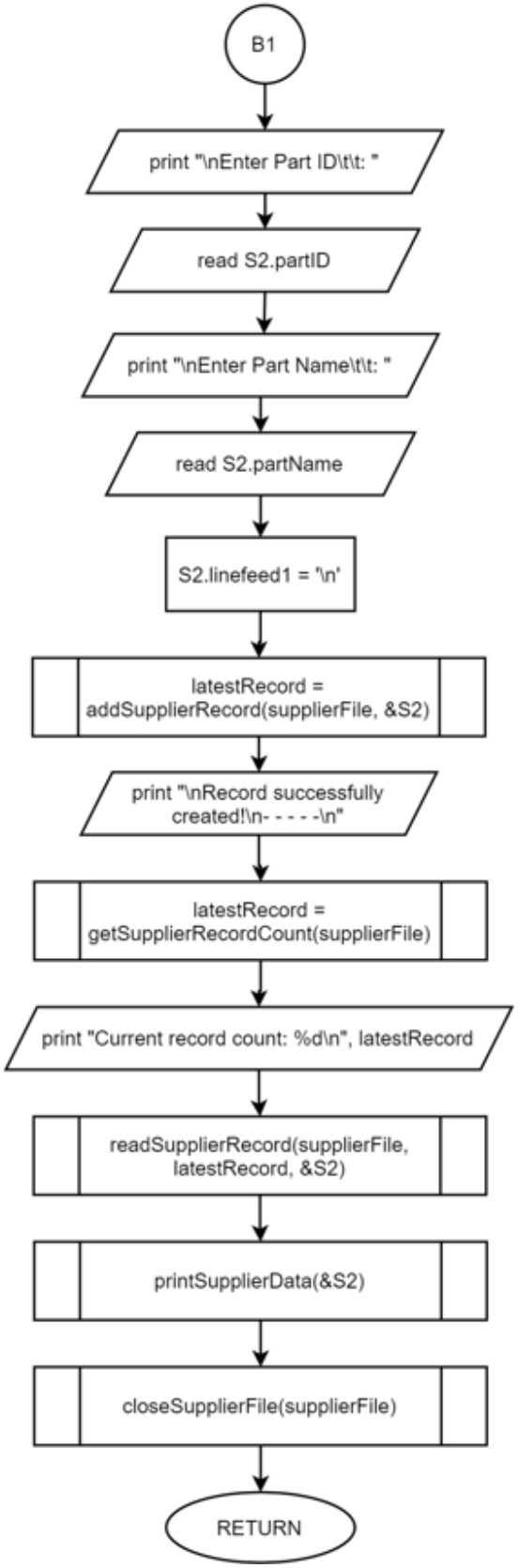

void partsInventoryCreation()
FILE* warehouseFile
partsInventory S1
DECLARE int latestRecord = 0
DECLARE int quantity = 0
DECLARE int convertquantity
DECLARE char warehouseCode[5]
print "1. Parts Inventory Creation in Warehouses\n"
memset(S1, 0 , sizeof(partsInventory))
print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nEnter Warehouse Code: "
read S1.warehouseCode
((S1.warehouseCode = "1") OR (S1.warehouseCode = "WBZ")) ?
False
True
S1.modelCode = "BZ"
S1.warehouseCode = "WBZ"
((S1.warehouseCode = "2") OR (S1.warehouseCode = "WSL"))?
False
True
S1.modelCode = "SL"
S1.warehouseCode = "WSL"
((S1.warehouseCode = "3") OR (S1.warehouseCode = "WAR"))?
True
False
S1.modelCode = "AR"
S1.warehouseCode = "WAR"
Print "\nInvalid warehouse code. Please try again!\n"
A1

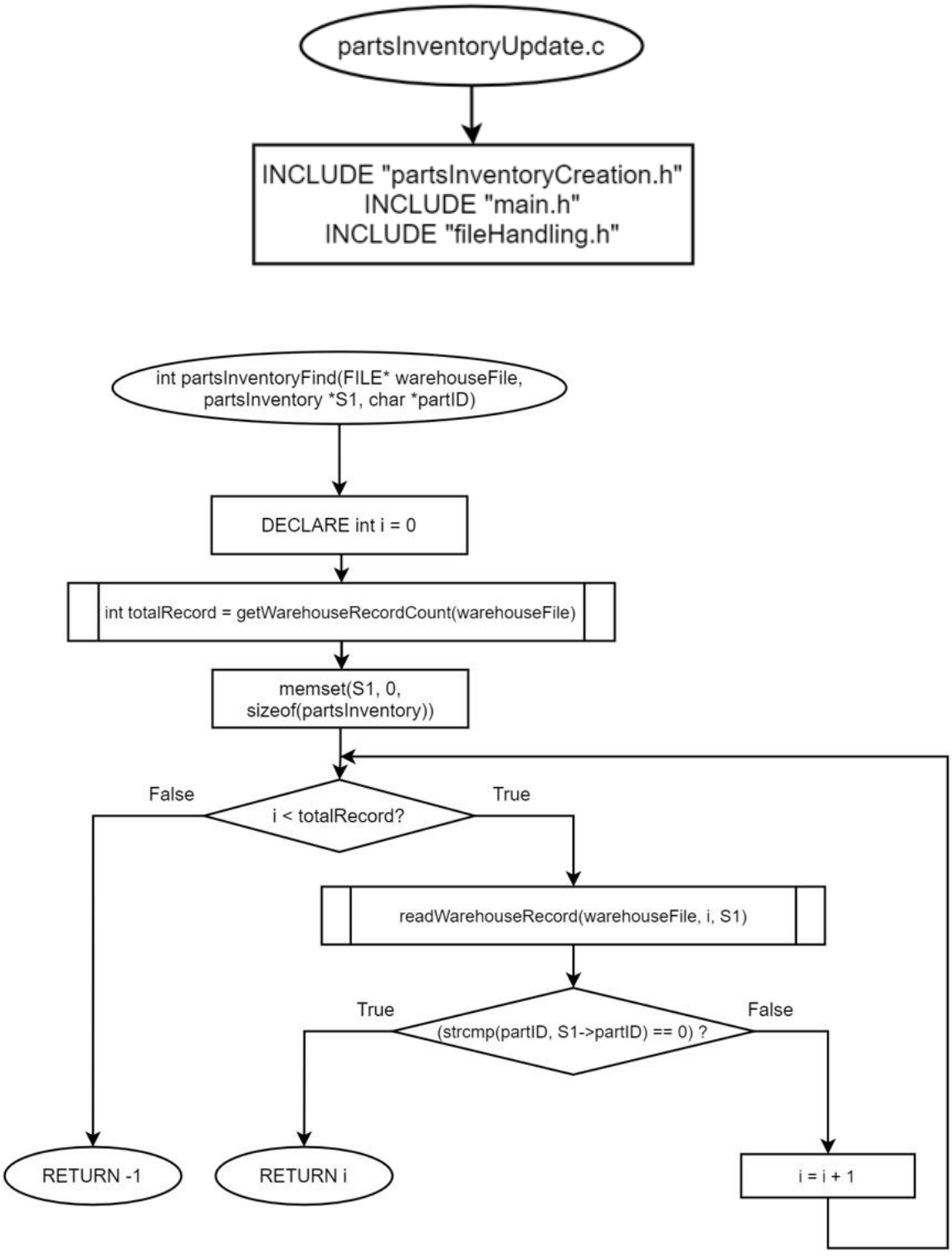

A1
warehouseFile = openWarehouseFile(S1.warehouseCode)
latestRecord = getWarehouseRecordCount(warehouseFile)
latestRecord = latestRecord + 1
stringprint(S1.partID, "%s%03d", warehouseCode, latestRecord)
Print "\nPart ID\t: %s%03d\n\n", warehouseCode, latestRecord
Print "\nEnter Section Code: "
Read S1.sectionCode
Print "Enter Part Name\t:
Read S1.partName
Print "Enter Quantity\t: "
Read S1.quantity
isInteger(S1.quantity)
(NOT isInteger(S1.quantity)?
False
True
Print "\nInput invalid, please try again with number\n"
S1.linefeed = '\n'
A2

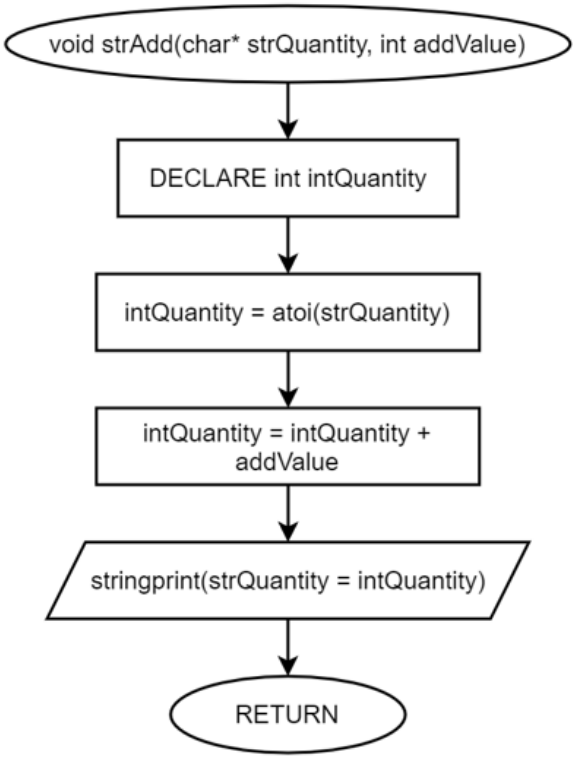

A2
latestRecord = addWarehouseRecord(warehouseFile, &S1)
Print "\n\Record successfully created!\n"
latestRecord = getWarehouseRecordCount(warehouseFile)
Print "Current record count: %d\n", latestRecord
readWarehouseRecord(warehouseFile, latestRecord, &S1)
printInventoryData(&S1)
closeWarehouseFile(warehouseFile)
RETURN

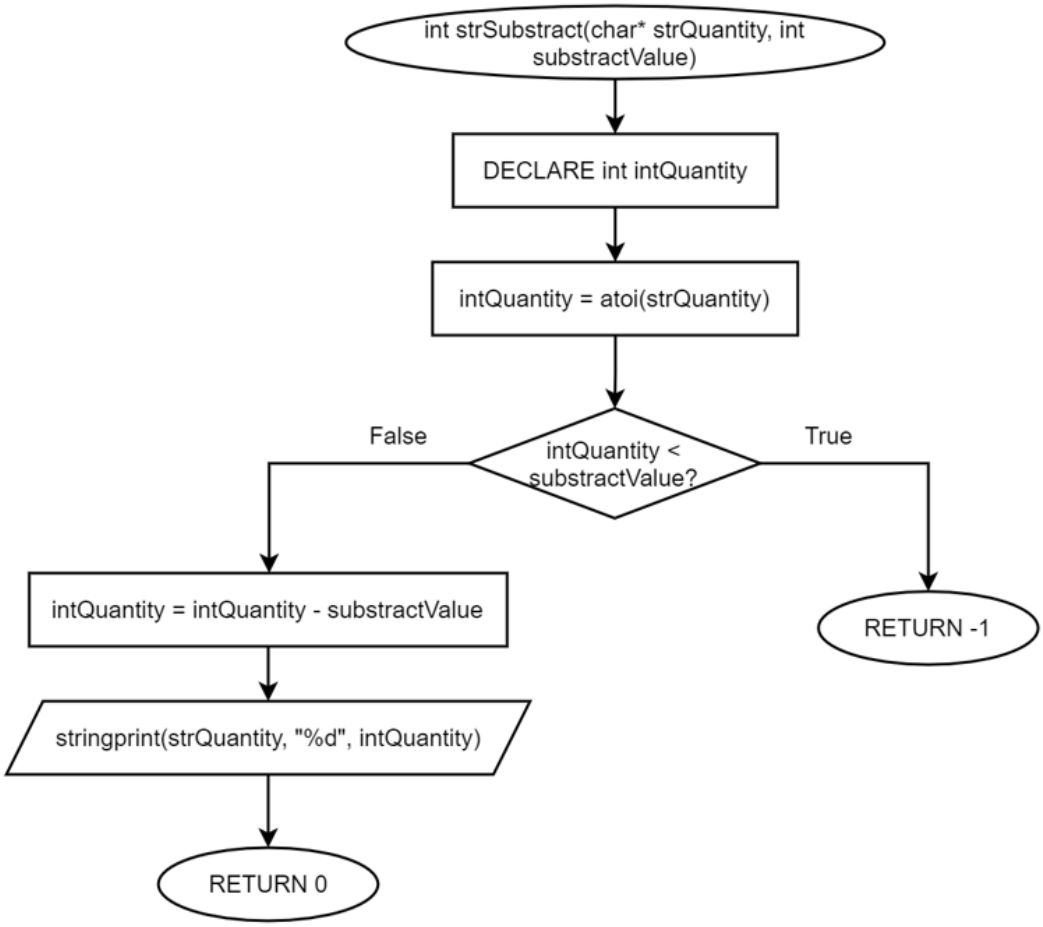

partsSupplier.c
INCLUDE "partsSupplier.h"
INCLUDE "fileHandling.h"
void printSupplierData(partsSupplier* sup)
print "Supplier ID\t: %s\n", sup->supplierID
print "Supplier Name\t: %s\n", sup->supplierName
print "Location\t: %s\n", sup->location
print "Part ID\t\t: %s\n", sup->partID
print "Part Name\t: %s\n", sup->partName
RETURN

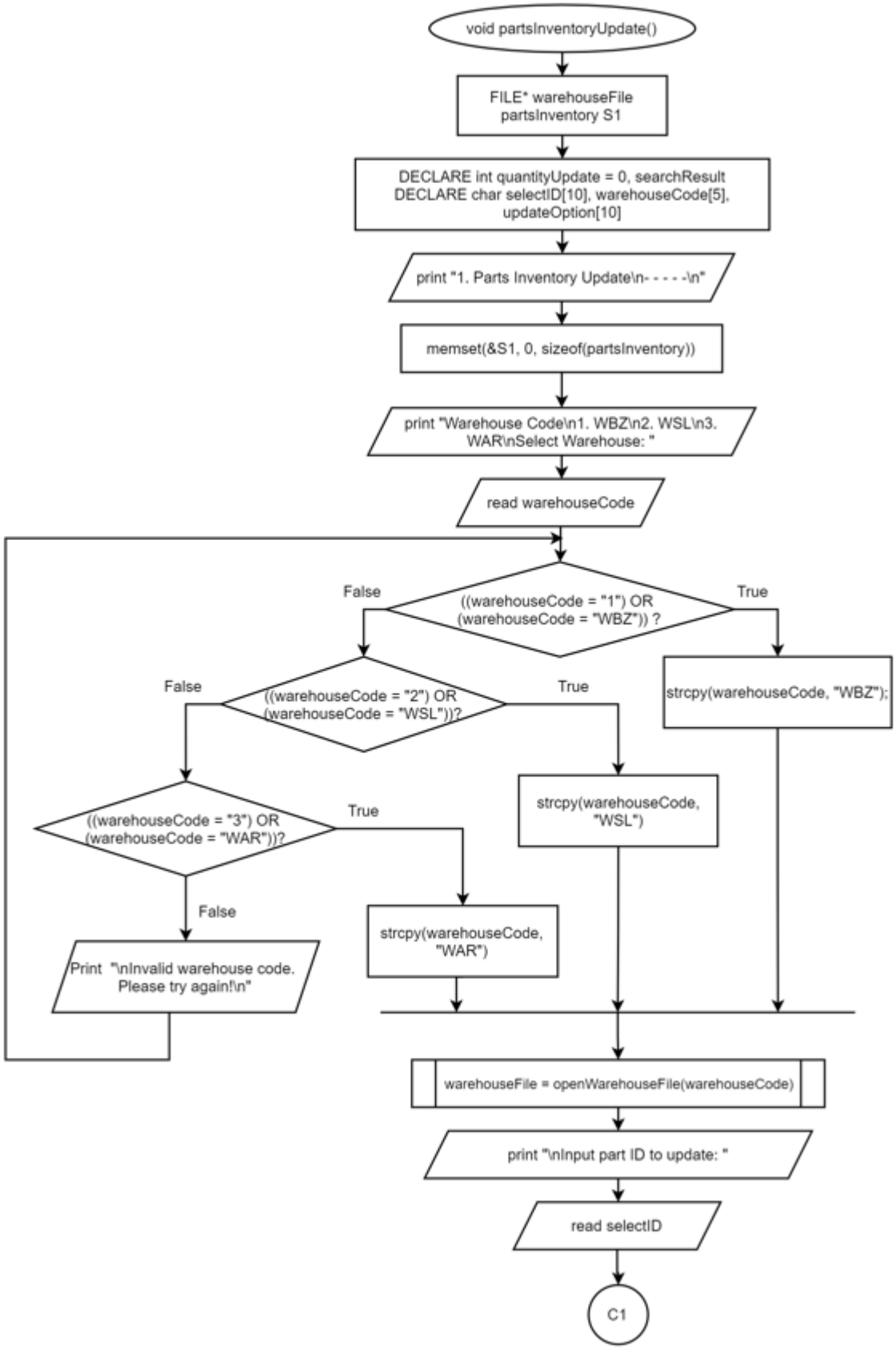

void partsSupplierCreation()
FILE* supplierFile
partsSupplier S2
DECLARE int latestRecord = 0
print"2. Parts Supplier Creation\n- - - - -\n"
memset(&S2, 0, sizeof(partsSupplier))
supplierFile = openSupplierFile()
print "*Input existing Supplier ID if there is any\nEnter Supplier ID\t: "
read S2.supplierID
print "\nEnter Supplier Name\t: "
read S2.supplierName
print "\nEnter location\t\t: "
read S2.location
print "\nEnter Part ID\t\t: "
read S2.partID
B1

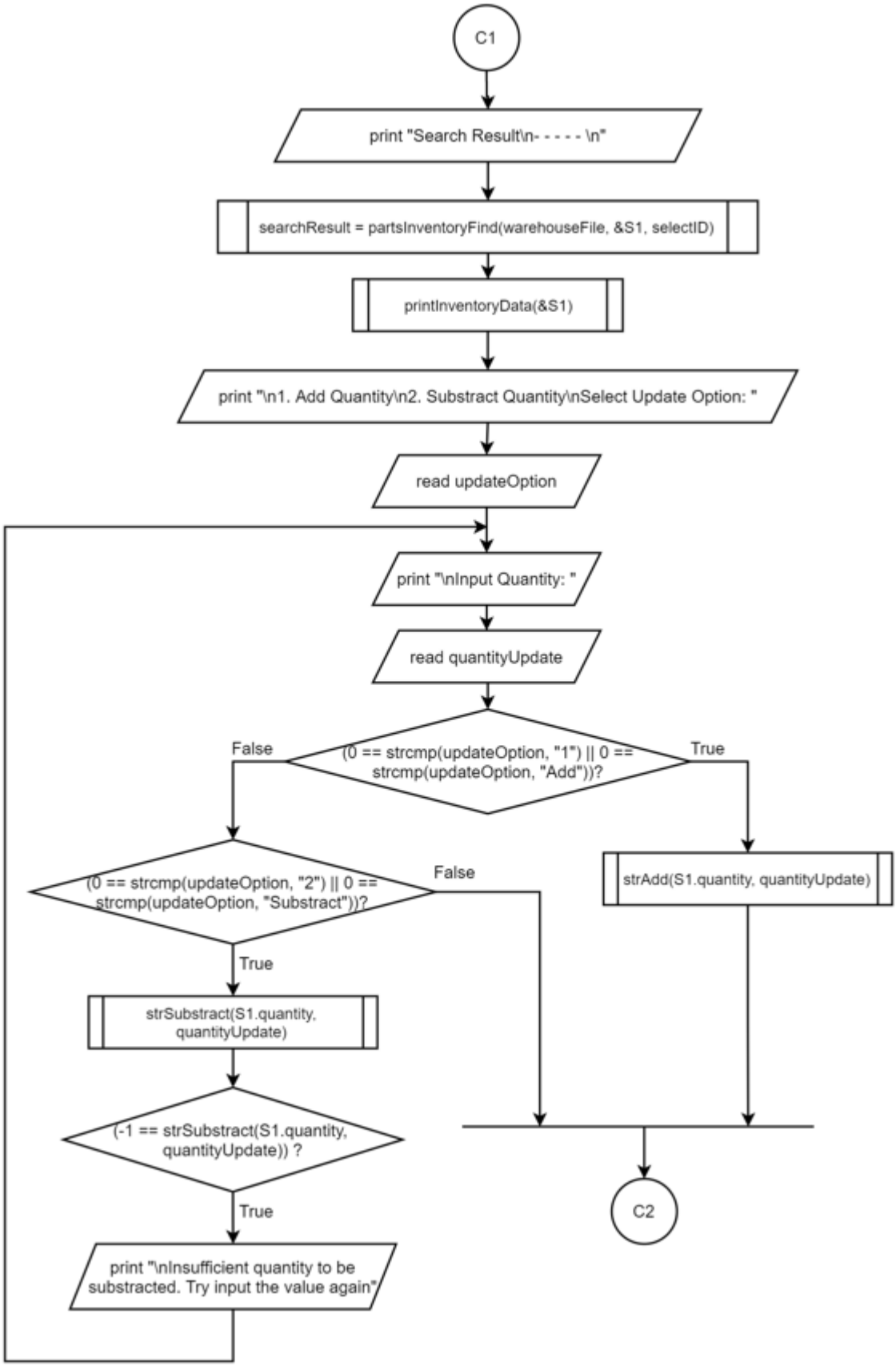

B1
print "\nEnter Part ID\t\t: "
read S2.partID
print "\nEnter Part Name\t\t: "
read S2.partName
S2.linefeed1 = '\n'
latestRecord = addSupplierRecord(supplierFile, &S2)
print "\nRecord successfully created!\n- - - - -\n"
latestRecord = getSupplierRecordCount(supplierFile)
print "Current record count: %d\n", latestRecord
readSupplierRecord(supplierFile, latestRecord, &S2)
printSupplierData(&S2)
closeSupplierFile(supplierFile)
RETURN

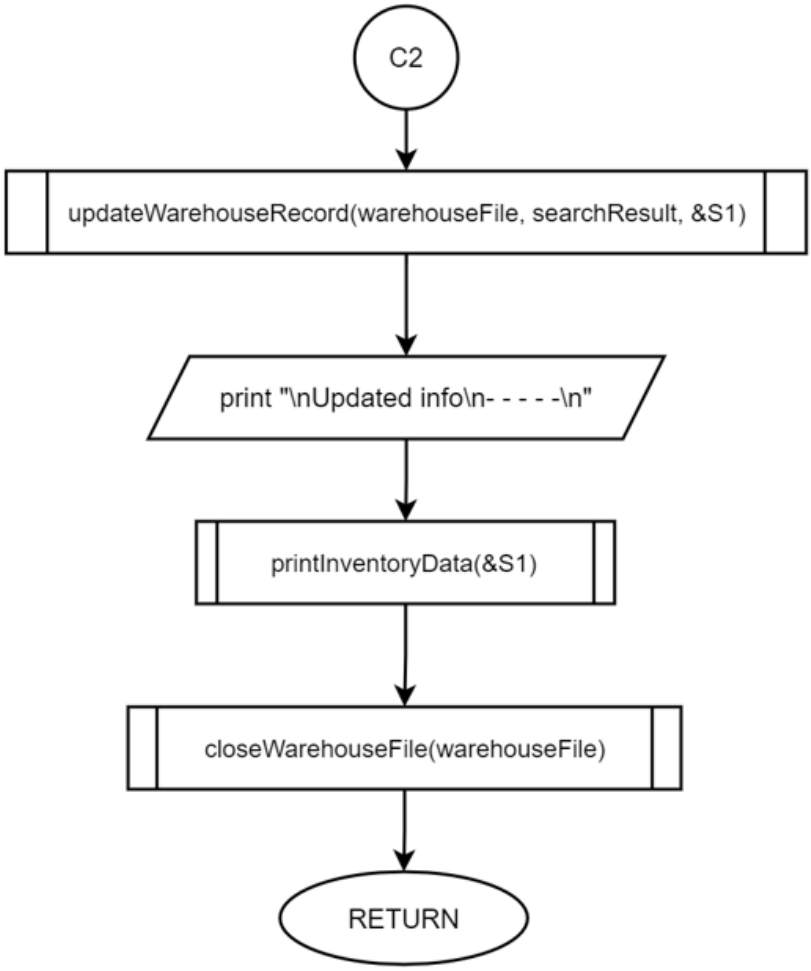

partsInventoryUpdate.c
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"
int partsInventoryFind(FILE* warehouseFile, partsInventory *S1, char *partID)
DECLARE int i = 0
int totalRecord = getWarehouseRecordCount(warehouseFile)
memset(S1, 0, sizeof(partsInventory))
False
i < totalRecord?
True
readWarehouseRecord(warehouseFile, i, S1)
True
(strcmp(partID, S1->partID) == 0) ?
False
RETURN -1
RETURN i
i = i + 1

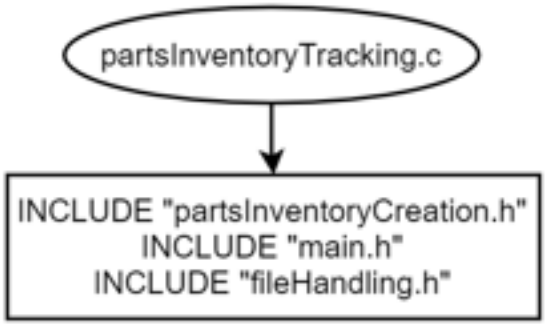

void strAdd(char* strQuantity, int addValue)
DECLARE int intQuantity
intQuantity = atoi(strQuantity)
intQuantity = intQuantity + addValue
stringprint(strQuantity = intQuantity)
RETURN

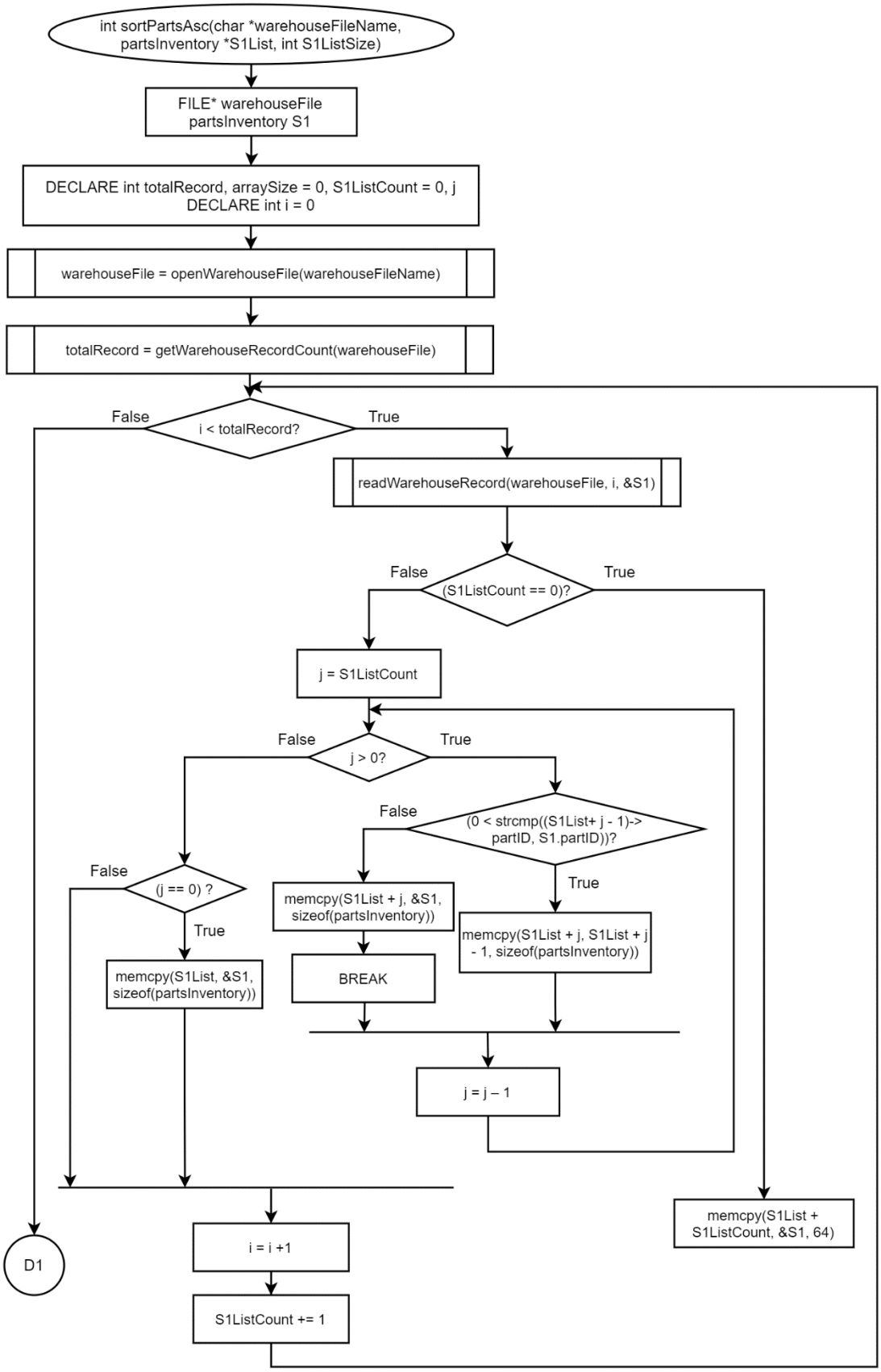

int strSubstract(char* strQuantity, int substractValue)
DECLARE int intQuantity
intQuantity = atoi(strQuantity)
intQuantity < substractValue?
False
True
intQuantity = intQuantity - substractValue
stringprint(strQuantity, "%d", intQuantity)
RETURN 0
RETURN -1

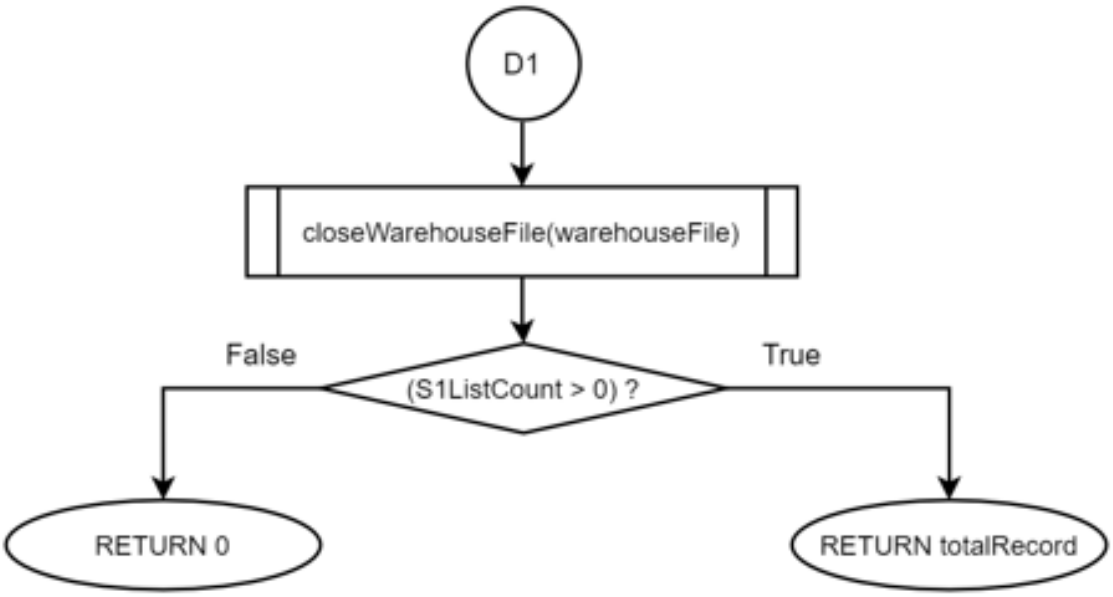

void partsInventoryUpdate()
FILE* warehouseFile
partsInventory S1
DECLARE int quantityUpdate = 0, searchResult
DECLARE char selectID[10], warehouseCode[5],
updateOption[10]
print "1. Parts Inventory Update\n- - - - -\n"
memset(&S1, 0, sizeof(partsInventory))
print "Warehouse Code\n1. WBZ\n2. WSL\n3.
WAR\nSelect Warehouse: "
read warehouseCode
((warehouseCode = "1") OR
(warehouseCode = "WBZ")) ?
False
True
strcpy(warehouseCode, "WBZ");
((warehouseCode = "2") OR
(warehouseCode = "WSL"))?
False
True
strcpy(warehouseCode,
"WSL")
((warehouseCode = "3") OR
(warehouseCode = "WAR"))?
True
False
strcpy(warehouseCode,
"WAR")
Print "\nInvalid warehouse code.
Please try again!\n"
warehouseFile = openWarehouseFile(warehouseCode)
print "\nInput part ID to update: "
read selectID
C1

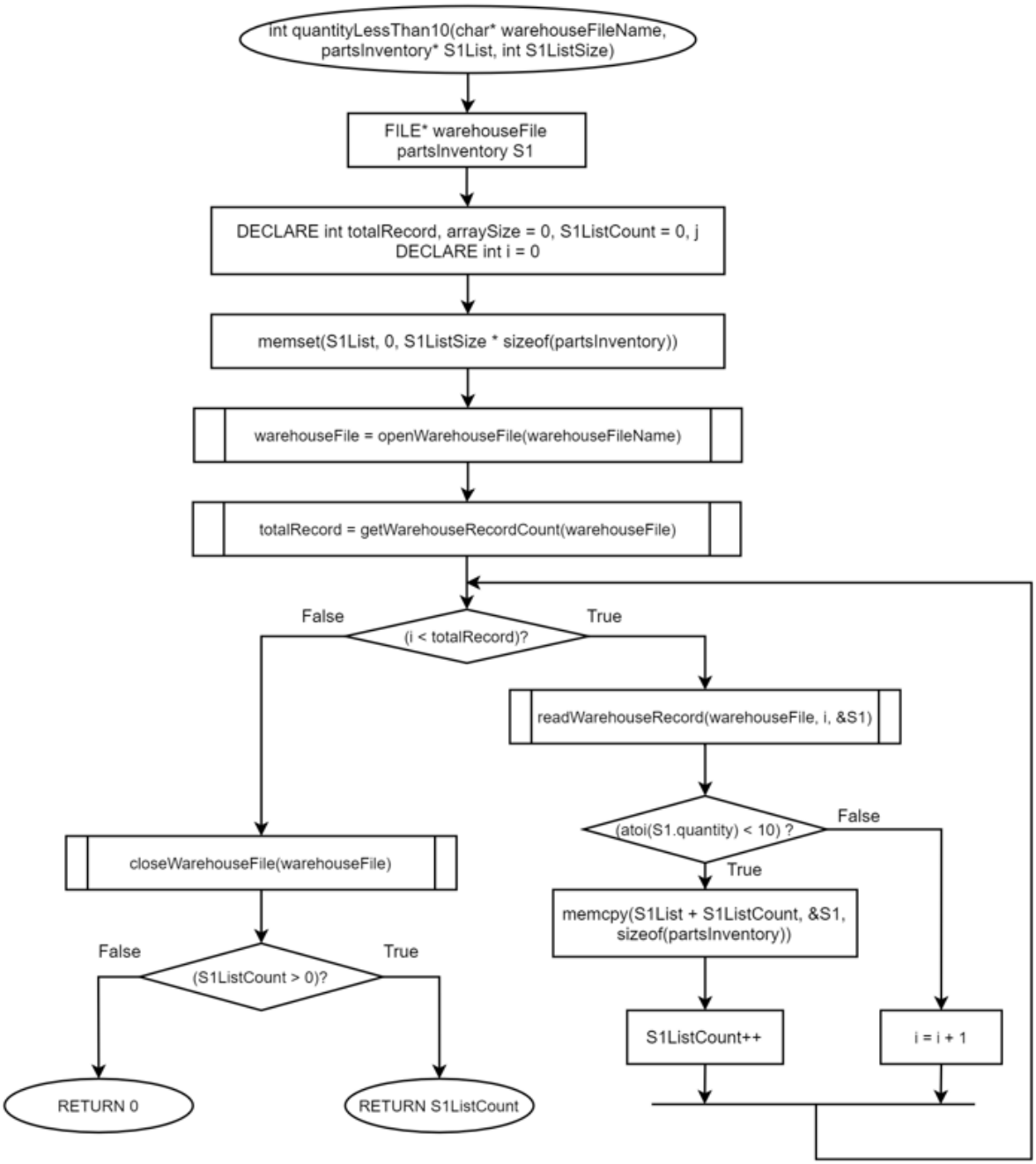

C1
print "Search Result\n- - - - - \n"
searchResult = partsInventoryFind(warehouseFile, &S1, selectID)
printInventoryData(&S1)
print "\n1. Add Quantity\n2. Substract Quantity\nSelect Update Option: "
read updateOption
print "\nInput Quantity: "
read quantityUpdate
(0 == strcmp(updateOption, "1") || 0 == strcmp(updateOption, "Add"))?
False
True
(0 == strcmp(updateOption, "2") || 0 == strcmp(updateOption, "Substract"))?
False
True
strAdd(S1.quantity, quantityUpdate)
strSubstract(S1.quantity, quantityUpdate)
(-1 == strSubstract(S1.quantity, quantityUpdate)) ?
True
print "\nInsufficient quantity to be substracted. Try input the value again"
C2

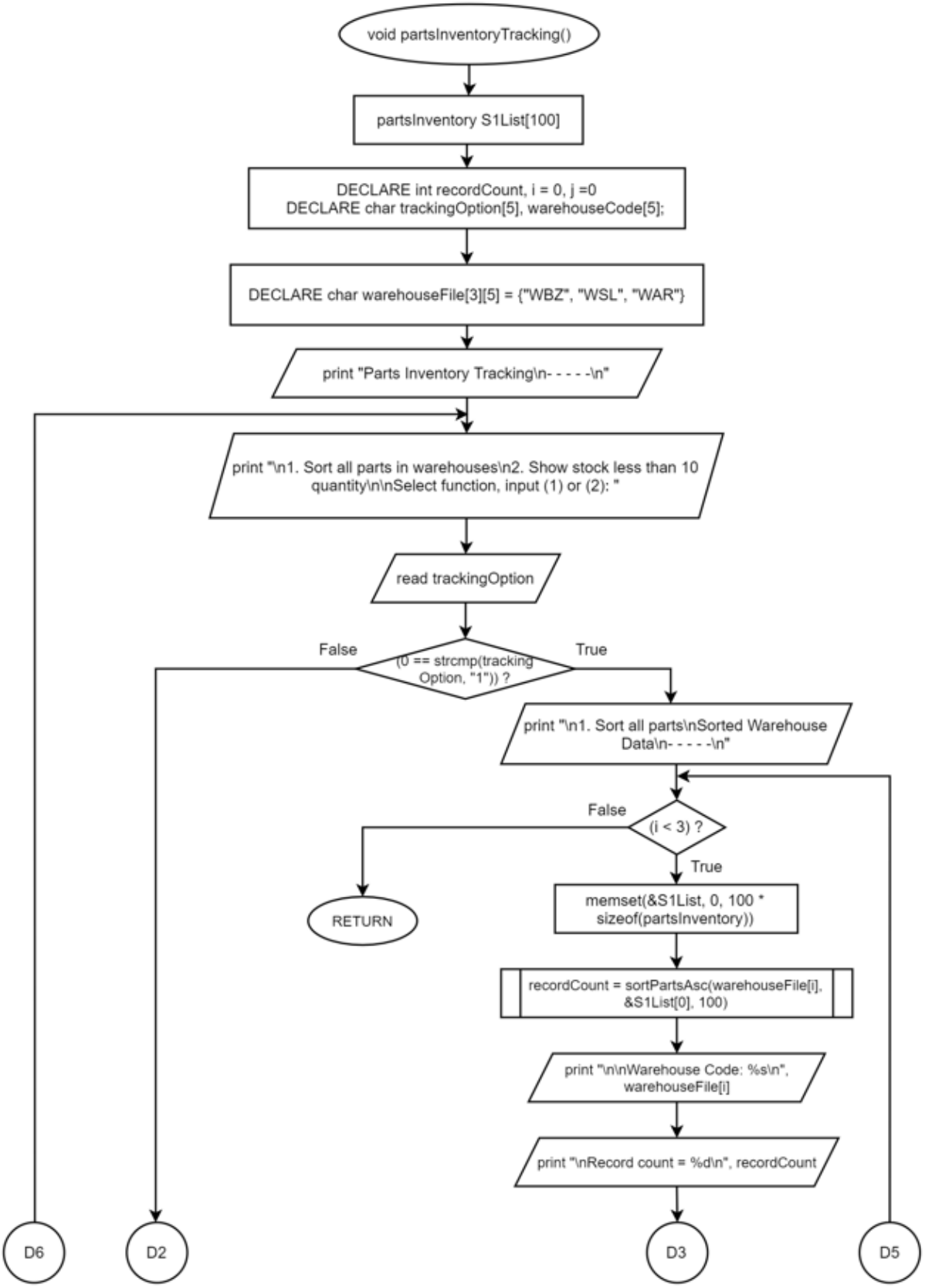

C2
updateWarehouseRecord(warehouseFile, searchResult, &S1)
print "\nUpdated info\n- - - - -\n"
printInventoryData(&S1)
closeWarehouseFile(warehouseFile)
RETURN

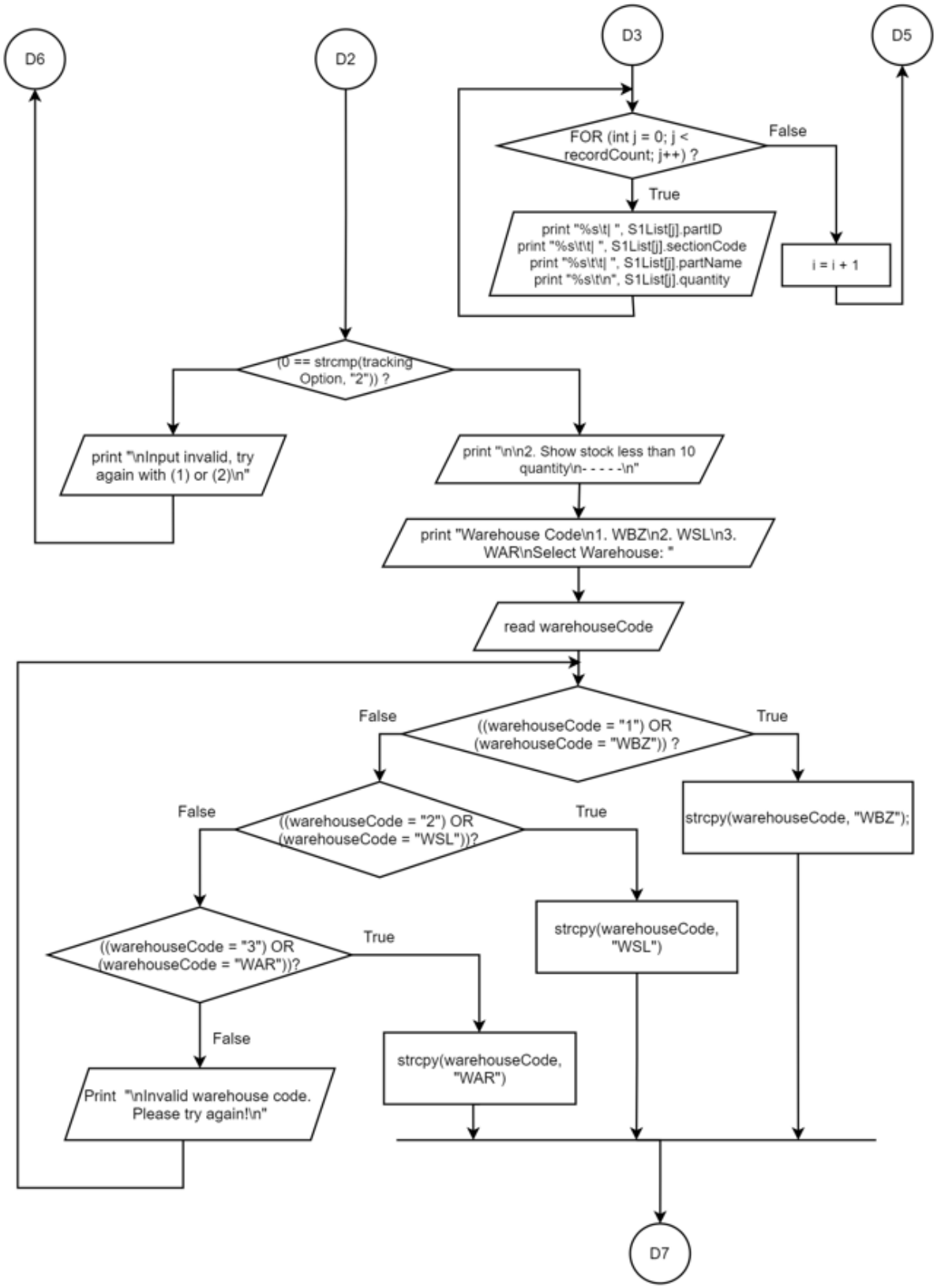

partsInventoryTracking.c
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"

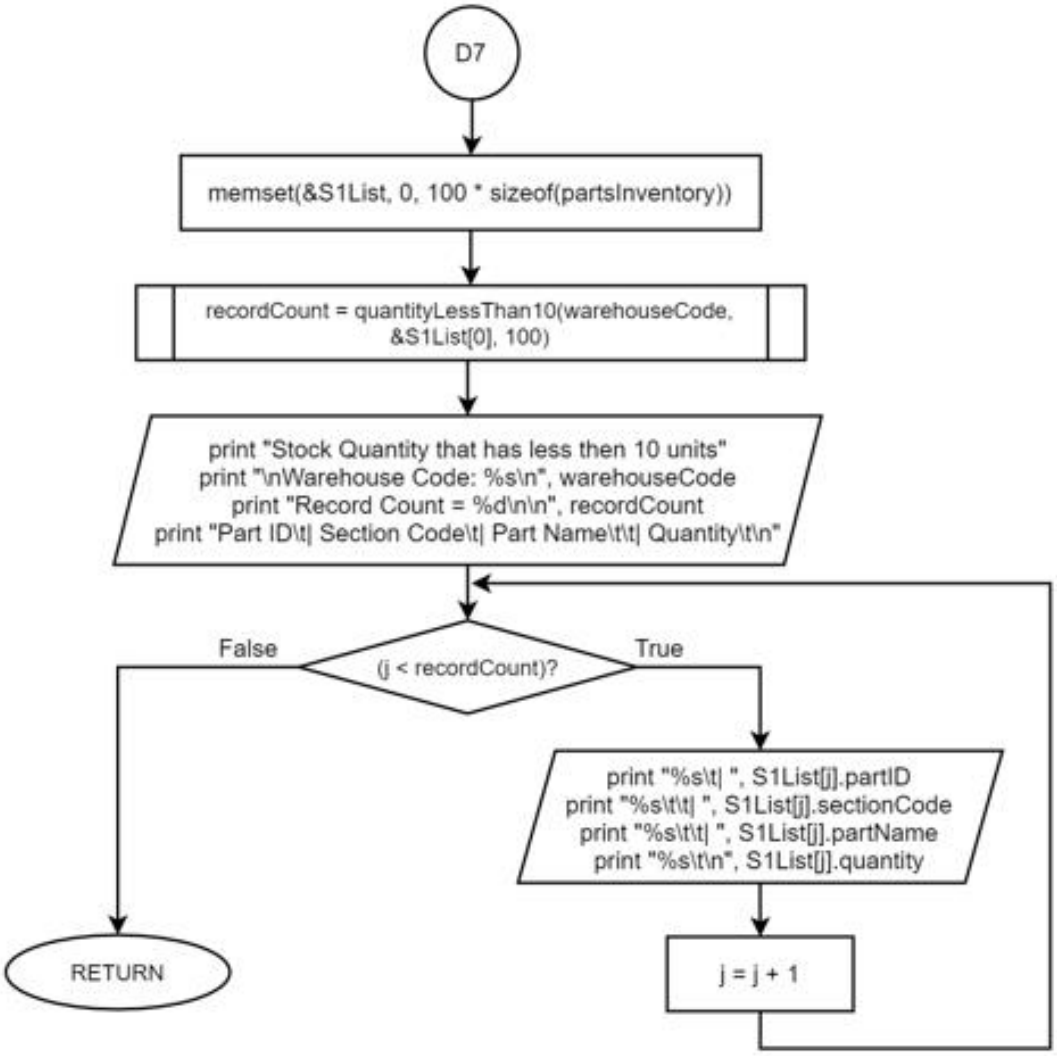

int sortPartsAsc(char *warehouseFileName, partsInventory *S1List, int S1ListSize)
FILE* warehouseFile partsInventory S1
DECLARE int totalRecord, arraySize = 0, S1ListCount = 0, j
DECLARE int i = 0
warehouseFile = openWarehouseFile(warehouseFileName)
totalRecord = getWarehouseRecordCount(warehouseFile)
False
i < totalRecord?
True
readWarehouseRecord(warehouseFile, i, &S1)
False
(S1ListCount == 0)?
True
j = S1ListCount
False
j > 0?
True
False
(0 < strcmp((S1List+ j - 1)-> partID, S1.partID))?
True
False
(j == 0) ?
True
memcpy(S1List + j, &S1, sizeof(partsInventory))
memcpy(S1List + j, S1List + j - 1, sizeof(partsInventory))
memcpy(S1List, &S1, sizeof(partsInventory))
BREAK
j = j – 1
memcpy(S1List + S1ListCount, &S1, 64)
D1
i = i +1
S1ListCount += 1

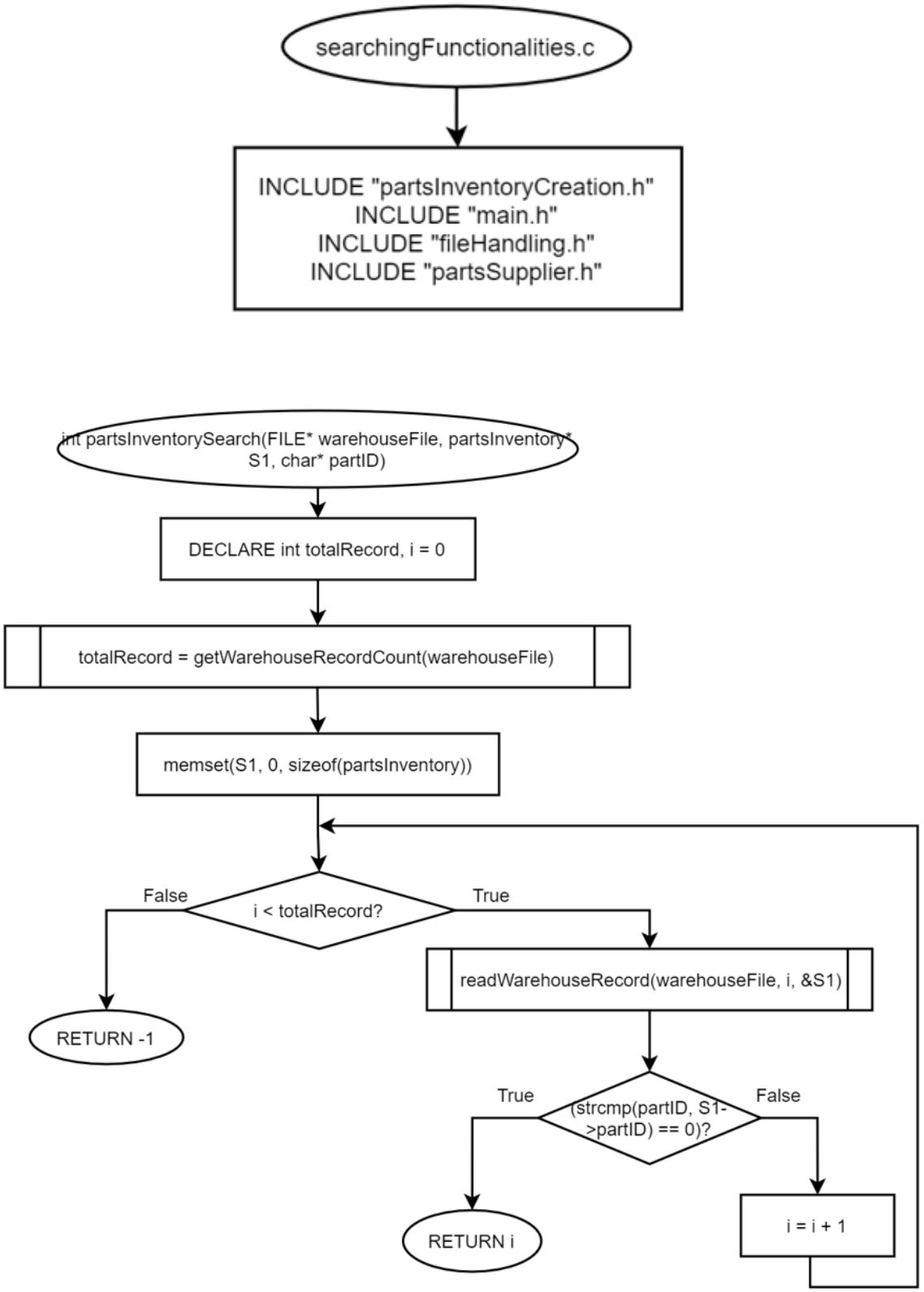

D1
closeWarehouseFile(warehouseFile)
False
(S1ListCount > 0) ?
True
RETURN 0
RETURN totalRecord

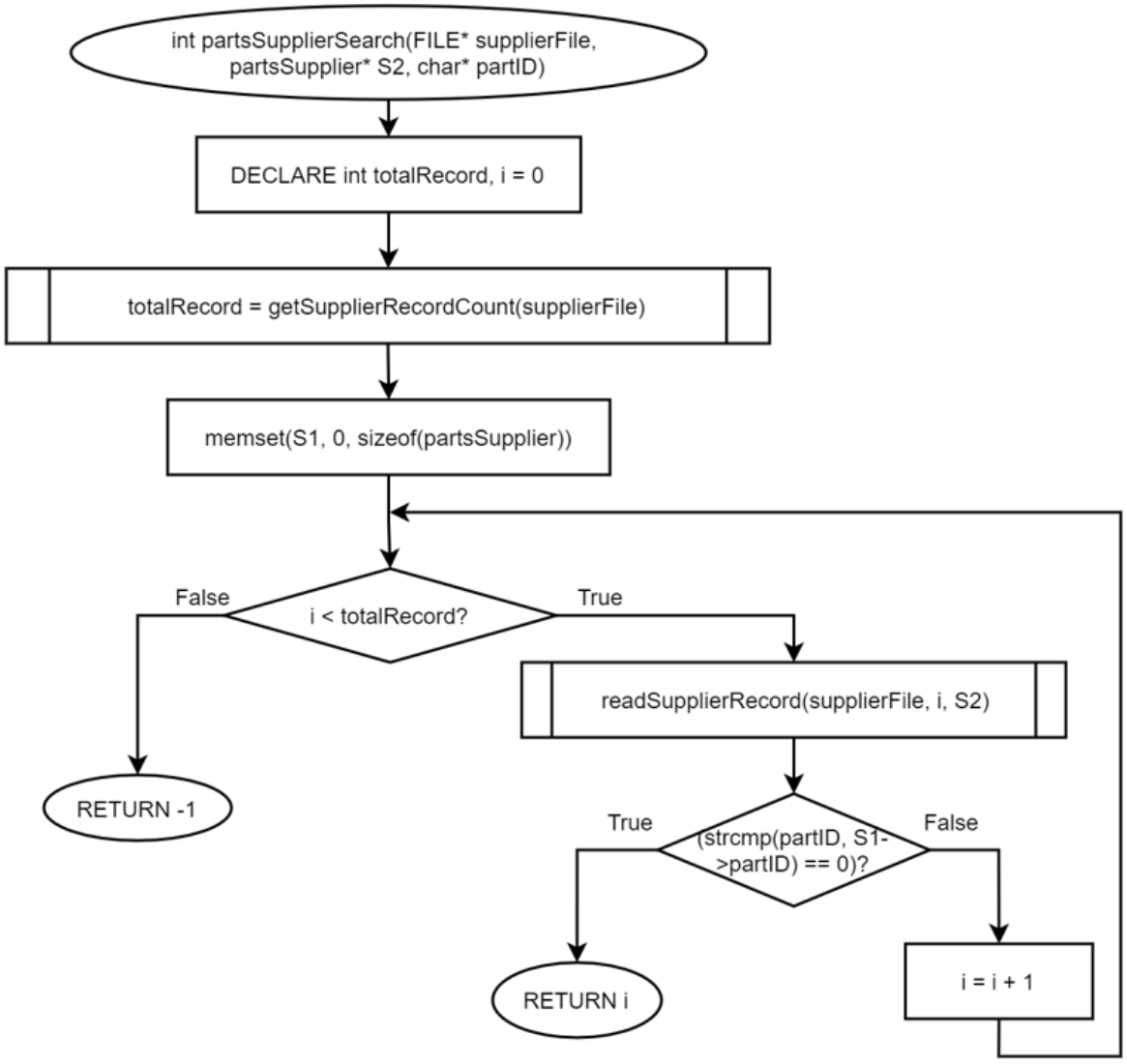

int quantityLessThan10(char* warehouseFileName, partsInventory* S1List, int S1ListSize)
FILE* warehouseFile
partsInventory S1
DECLARE int totalRecord, arraySize = 0, S1ListCount = 0, j
DECLARE int i = 0
memset(S1List, 0, S1ListSize * sizeof(partsInventory))
warehouseFile = openWarehouseFile(warehouseFileName)
totalRecord = getWarehouseRecordCount(warehouseFile)
False
(i < totalRecord)?
True
readWarehouseRecord(warehouseFile, i, &S1)
(atoi(S1.quantity) < 10) ?
False
True
memcpy(S1List + S1ListCount, &S1, sizeof(partsInventory))
S1ListCount++
i = i + 1
closeWarehouseFile(warehouseFile)
False
(S1ListCount > 0)?
True
RETURN 0
RETURN S1ListCount

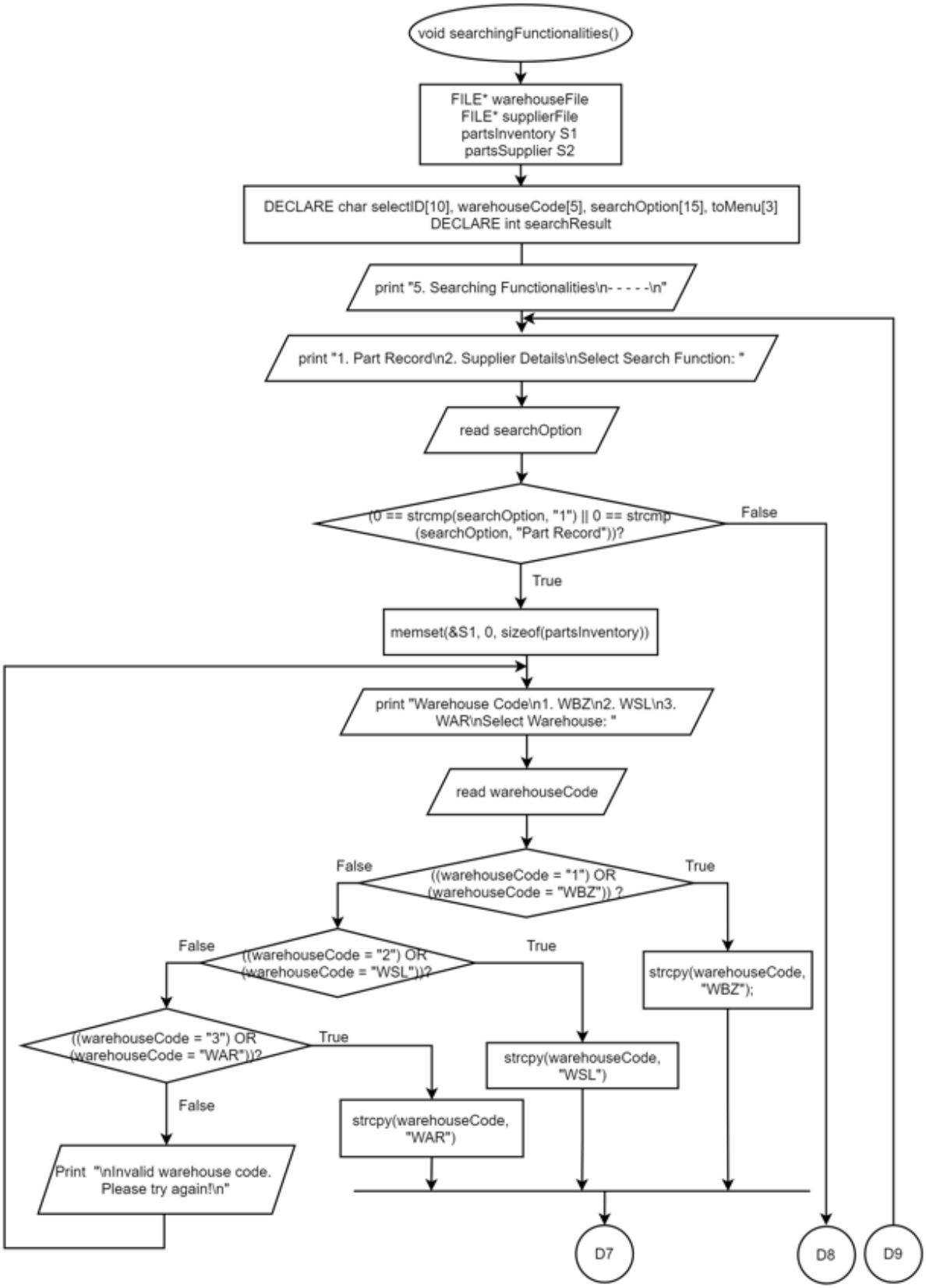

void partsInventoryTracking()
partsInventory S1List[100]
DECLARE int recordCount, i = 0, j =0
DECLARE char trackingOption[5], warehouseCode[5];
DECLARE char warehouseFile[3][5] = {"WBZ", "WSL", "WAR"}
print "Parts Inventory Tracking\n- - - - -\n"
print "\n1. Sort all parts in warehouses\n2. Show stock less than 10 quantity\n\nSelect function, input (1) or (2): "
read trackingOption
False
(0 == strcmp(tracking Option, "1")) ?
True
print "\n1. Sort all parts\nSorted Warehouse Data\n- - - - -\n"
False
(i < 3) ?
True
RETURN
memset(&S1List, 0, 100 * sizeof(partsInventory))
recordCount = sortPartsAsc(warehouseFile[i], &S1List[0], 100)
print "\n\nWarehouse Code: %s\n", warehouseFile[i]
print "\nRecord count = %d\n", recordCount
D6
D2
D3
D5

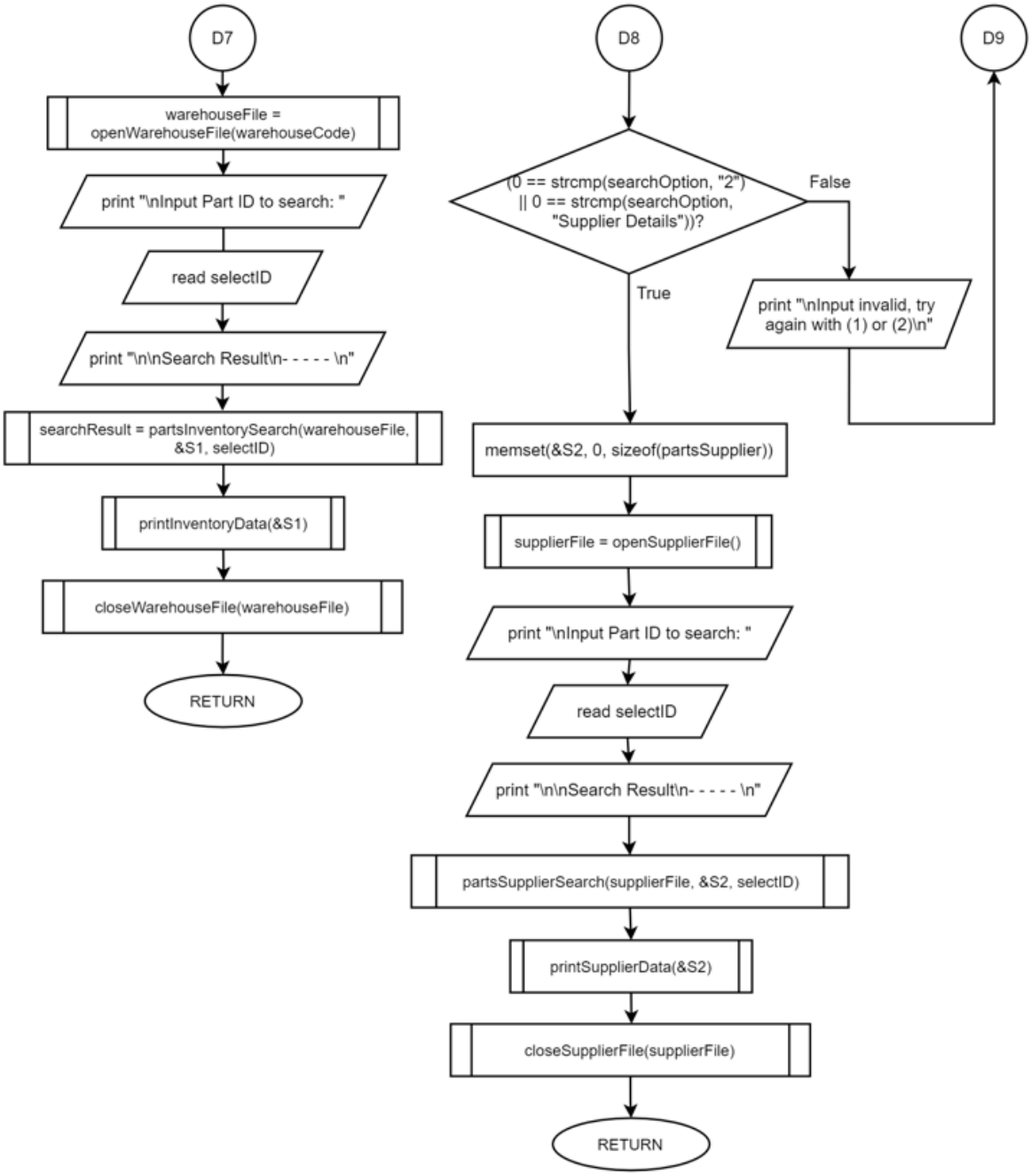

D6
D2
D3
D5
FOR (int j = 0; j < recordCount; j++) ?
False
True
print "%s\t| ", S1List[j].partID
print "%s\t\t| ", S1List[j].sectionCode
print "%s\t\t| ", S1List[j].partName
print "%s\t\n", S1List[j].quantity
i = i + 1
(0 == strcmp(tracking Option, "2")) ?
print "\nInput invalid, try again with (1) or (2)\n"
print "\n\n2. Show stock less than 10 quantity\n- - - - -\n"
print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: "
read warehouseCode
((warehouseCode = "1") OR (warehouseCode = "WBZ")) ?
False
True
strcpy(warehouseCode, "WBZ");
((warehouseCode = "2") OR (warehouseCode = "WSL"))?
False
True
strcpy(warehouseCode, "WSL")
((warehouseCode = "3") OR (warehouseCode = "WAR"))?
True
False
strcpy(warehouseCode, "WAR")
Print "\nInvalid warehouse code. Please try again!\n"
D7


D7
memset(&S1List, 0, 100 * sizeof(partsInventory))
recordCount = quantityLessThan10(warehouseCode, &S1List[0], 100)
print "Stock Quantity that has less then 10 units"
print "\nWarehouse Code: %s\n", warehouseCode
print "Record Count = %d\n\n", recordCount
print "Part ID\t| Section Code\t| Part Name\t\t| Quantity\t\n"
False
(j < recordCount)?
True
print "%s\t| ", S1List[j].partID
print "%s\t\t| ", S1List[j].sectionCode
print "%s\t\t| ", S1List[j].partName
print "%s\t\n", S1List[j].quantity
RETURN
j = j + 1


searchingFunctionalities.c
INCLUDE "partsInventoryCreation.h"
INCLUDE "main.h"
INCLUDE "fileHandling.h"
INCLUDE "partsSupplier.h"
int partsInventorySearch(FILE* warehouseFile, partsInventory* S1, char* partID)
DECLARE int totalRecord, i = 0
totalRecord = getWarehouseRecordCount(warehouseFile)
memset(S1, 0, sizeof(partsInventory))
i < totalRecord?
False
True
RETURN -1
readWarehouseRecord(warehouseFile, i, &S1)
(strcmp(partID, S1->partID) == 0)?
True
False
RETURN i
i = i + 1


int partsSupplierSearch(FILE* supplierFile, partsSupplier* S2, char* partID)
DECLARE int totalRecord, i = 0
totalRecord = getSupplierRecordCount(supplierFile)
memset(S1, 0, sizeof(partsSupplier))
i < totalRecord?
False
True
RETURN -1
readSupplierRecord(supplierFile, i, S2)
(strcmp(partID, S1->partID) == 0)?
True
False
RETURN i
i = i + 1


void searchingFunctionalities()
FILE* warehouseFile
FILE* supplierFile
partsInventory S1
partsSupplier S2
DECLARE char selectID[10], warehouseCode[5], searchOption[15], toMenu[3]
DECLARE int searchResult
print "5. Searching Functionalities\n- - - - -\n"
print "1. Part Record\n2. Supplier Details\nSelect Search Function: "
read searchOption
(0 == strcmp(searchOption, "1") || 0 == strcmp (searchOption, "Part Record"))?
False
True
memset(&S1, 0, sizeof(partsInventory))
print "Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: "
read warehouseCode
False
((warehouseCode = "1") OR (warehouseCode = "WBZ")) ?
True
False
((warehouseCode = "2") OR (warehouseCode = "WSL"))?
True
strcpy(warehouseCode, "WBZ");
((warehouseCode = "3") OR (warehouseCode = "WAR"))?
True
strcpy(warehouseCode, "WSL")
False
strcpy(warehouseCode, "WAR")
Print "\nInvalid warehouse code. Please try again!\n"
D7
D8
D9


D7
warehouseFile = openWarehouseFile(warehouseCode)
print "\nInput Part ID to search: "
read selectID
print "\n\nSearch Result\n- - - - - \n"
searchResult = partsInventorySearch(warehouseFile, &S1, selectID)
printInventoryData(&S1)
closeWarehouseFile(warehouseFile)
RETURN
D8
(0 == strcmp(searchOption, "2") || 0 == strcmp(searchOption, "Supplier Details"))?
False
True
print "\nInput invalid, try again with (1) or (2)\n"
D9
memset(&S2, 0, sizeof(partsSupplier))
supplierFile = openSupplierFile()
print "\nInput Part ID to search: "
read selectID
print "\n\nSearch Result\n- - - - - \n"
partsSupplierSearch(supplierFile, &S2, selectID)
printSupplierData(&S2)
closeSupplierFile(supplierFile)
RETURN

3.0 Program Source Code

3.1 Header file

main.h

It contains the preprocessor directive that required for the program.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
```

menu.h

It contains declaration of shared function from menu.c.

```
#ifndef menu_header
#define menu_header

void menu();

#endif
```

fileHandling.h

It includes shared functions from other header files and also declare shared function of fileHandling.c.

```
#ifndef fileHandling_Header
#define fileHandling_Header

#include "partsSupplier.h"
#include "partsInventoryCreation.h"

FILE* openWarehouseFile(char* warehouseCode);
int readWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData);
int addWarehouseRecord(FILE* fileHandle, partsInventory* partData);
int updateWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData);
int getWarehouseRecordCount(FILE* fileHandle);
void closeWarehouseFile(FILE* fileHandle);

FILE* openSupplierFile();
int readSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData);
int addSupplierRecord(FILE* fileHandle, partsSupplier* supplierData);
int updateSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData);
int getSupplierRecordCount(FILE* fileHandle);
void closeSupplierFile(FILE* fileHandle);

#endif
```

partsInventoryCreation.h

This header file contains partsInventory structure and declare shared function of fileHandling.c.

```c
#ifndef partInventoryCreation_Header
#define partInventoryCreation_Header

typedef struct PARTSINVENTORY_ST{
    char partID[10],
        sectionCode[5],
        partName[20],
        quantity[5],
        linefeed;
} partsInventory;

void partsInventoryCreation();
void printInventoryData(partsInventory* inv);
int isInteger(char* checkData);

#endif
```

partsSupplier.h

It contains partsSupplier structure and the declaration of shared function of partSupplier.c.

```c
#ifndef partSupplier_Header
#define partSupplier_Header

typedef struct {
    char supplierID[10],
        supplierName[20],
        location[16],
        partID[10],
        partName[20],
        linefeed1;
} partsSupplier;

void printSupplierData(partsSupplier* sup);
void partsSupplierCreation();

#endif
```

partsInventoryUpdate.h

It has a declaration of shared function from partsInventory.c.

```
1  #ifndef partInventoryUpdate_Header
2  #define partInventoryUpdate_Header
3
4  void partsInventoryUpdate();
5
6  #endif
```

partsInventoryTracking.h

It has a declaration of shared function from partsInventoryTracking.c.

```
1  #ifndef partsInventoryTracking_header
2  #define partsInventoryTracking_header
3
4  void partsInventoryTracking();
5
6  #endif
```

searchingFunctionalities.h

It has a declaration of shared function from searchingFunctionalities.c.

```
1  #ifndef searchingFunctionalities_Header
2  #define searchingFunctionalities_Header
3
4  void searchingFunctionalities();
5
6  #endif
```

3.2 Source File

main.c

This source file contains a loop that allows the user to stay in the program. The user may input 'x' to end the program. Before completely exit from the program, the system will prompt out a goodbye message to the user.

```c
#include "main.h"
#include "menu.h"

int main(){
    char cont;

    //Loop menu until user wants to end the program
    do {
        printf("Welcome to Automobile Parts Inventory Management System\n- - - - -\n");
        menu();

        printf("\nPress <enter> to continue OR 'X' to end: ");
        scanf("%c", &cont);
        system("cls");
    } while ((cont != 'x') && (cont != 'X'));

    printf("\nThank you and have a great day! \n");
    return 0;
}
```

menu.c

This C file act as the main menu that able to call the other functions by identifying the user menu option through switch statement.

```c
#include "partsInventoryCreation.h"
#include "partsSupplier.h"
#include "partsInventoryTracking.h"
#include "partsInventoryUpdate.h"
#include "searchingFunctionalities.h"

void menu(){
    printf("1. Parts Inventory Creation in Warehouses\n2. Parts Supplier Creation\n3. Parts Inventory Update\n");
    printf("4. Parts Inventory Tracking\n5. Searching Functionalities\n6. Exit\n");
    int option;

    //User select menu option
    printf("\nMenu Option: ");
    scanf("%d", &option);
    getchar();
    printf("\n");


    //Switch case to call function
    switch (option){
        case 1:
            system("cls");
            partsInventoryCreation();
            break;
        case 2:
            system("cls");
            partsSupplierCreation();
            break;
        case 3:
            system("cls");
            partsInventoryUpdate();
            break;
        case 4:
            system("cls");
            partsInventoryTracking();
            break;
        case 5:
            system("cls");
            searchingFunctionalities();
            system("pause");
            break;
        case 6:
            break;
        default:
            printf("\nInvalid menu selection, select only from (1) to (6)\n\n");
            system("pause");
            break;
    }
}
```

fileHandling.c

fileHandling.c is crucial in this program as it contains all the functions from open file, read file, write file, update file, close file and more. All the functions that were written in it, are able to be used in various situation that involves with parts inventory and supplier record.

```c
#include "main.h"
#include "partsInventoryCreation.h"
#include "partsSupplier.h"

//open warehouse file
FILE* openWarehouseFile(char* warehouseCode) {
    char fileName[128] = { 0 };
    FILE* fileHandle = 0;

    //construct file name using warehouseCode
    sprintf(fileName, "%sinventory.txt", warehouseCode);

    //read and create the file
    fileHandle = fopen(fileName, "r+b");
    //write and create the file
    if (0 == fileHandle) {
        fileHandle = fopen(fileName, "w+b");
    }
    return fileHandle;
}

//read for specific warehouse record
int readWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData) {

    //read fail
    if (0 > fseek(fileHandle, recordID * sizeof(partsInventory), SEEK_SET))
        return -1;

    //read file
    fread(partData, sizeof(partsInventory), 1, fileHandle);

    return 0;
}
```

```c
//add warehouse record at the end of the file
int addWarehouseRecord(FILE* fileHandle, partsInventory* partData) {
    int s = 0;
    fseek(fileHandle, 0L, SEEK_END);

    s = fwrite(partData, sizeof(partsInventory), 1, fileHandle);

    return (sizeof(partsInventory) == s);
}

//update warehouse record on the original line
int updateWarehouseRecord(FILE* fileHandle, int recordID, partsInventory* partData) {
    int writeSize = 0;

    if (0 > fseek(fileHandle, recordID * sizeof(partsInventory), SEEK_SET))
        return -1;
    writeSize = fwrite(partData, sizeof(partsInventory), 1, fileHandle);
    fflush(fileHandle);
    return 0;
}

//get the total warehouse record count
int getWarehouseRecordCount(FILE* fileHandle) {
    fseek(fileHandle, 0L, SEEK_END);
    return (ftell(fileHandle) / sizeof(partsInventory));
}

//close warehouse file
void closeWarehouseFile(FILE* fileHandle)
{
    fclose(fileHandle);
}
```

```c
//open supplier file for append
FILE* openSupplierFile() {
    char fileName[128] = { 0 };
    FILE* fileHandle = 0;

    fileHandle = fopen("Supplier.txt", "a+b");
    return fileHandle;
}

//read for specific supplier record
int readSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData) {

    //read fail
    if (0 > fseek(fileHandle, recordID * sizeof(partsSupplier), SEEK_SET))
        return -1;

    fread(supplierData, sizeof(partsSupplier), 1, fileHandle);

    return 0;
}

//add supplier record at the end of the file
int addSupplierRecord(FILE* fileHandle, partsSupplier* supplierData) {
    int s = 0;
    fseek(fileHandle, 0L, SEEK_END);

    s = fwrite(supplierData, sizeof(partsSupplier), 1, fileHandle);

    return (sizeof(partsSupplier) == s);
}
```

```c
//update supplier record at the orginal line
int updateSupplierRecord(FILE* fileHandle, int recordID, partsSupplier* supplierData) {
    if (0 > fseek(fileHandle, recordID * sizeof(partsSupplier), SEEK_SET))
        return -1;

    return (sizeof(partsSupplier) == fwrite(supplierData, sizeof(partsSupplier), 1, fileHandle));
}

//get total supplier record count
int getSupplierRecordCount(FILE* fileHandle) {
    fseek(fileHandle, 0L, SEEK_END);
    return (ftell(fileHandle) / sizeof(partsSupplier));
}

//close supplier file
void closeSupplierFile(FILE* fileHandle)
{
    fclose(fileHandle);
}
```

partsInventoryCreation.c

This source file allows the user to create a new part inventory record and store it in text files.

```c
#include "partsInventoryCreation.h"
#include "main.h"
#include "fileHandling.h"

//to print parts inventory structure
void printInventoryData(partsInventory* inv)
{
    printf("Part ID\t\t: %s\n", inv->partID);
    printf("Section Code\t: %s\n", inv->sectionCode);
    printf("Part Name\t: %s\n", inv->partName);
    printf("Quantity\t: %s\n", inv->quantity);
}

//check whether input char is integer
int isInteger(char* checkData)
{
    if (0 == checkData)
        return 0;
    if (checkData[0] == '0' && strlen(checkData) == 1)
        return 1;
    if (0 == atoi(checkData))
        return 0;

    return 1;
}
```

```c
//main function
void partsInventoryCreation()
{
    //declare pointers, structure and variable
    FILE* warehouseFile;
    partsInventory S1;
    int quantity = 0;
    int latestRecord = 0;
    char warehouseCode[5];
    int convertquantity;

    printf("1. Parts Inventory Creation in Warehouses\n- - - - -\n");

reinputwarehouse:
    //reset structure memory
    memset(&S1, 0, sizeof(partsInventory));
    printf("Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: ");
    scanf("%s", warehouseCode);
    getchar();

    //set warehouseCode as the user input
    if (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) {
        strcpy(warehouseCode, "WBZ");
    }
    else if (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL")) {
        strcpy(warehouseCode, "WSL");
    }
    else if (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR")) {
        strcpy(warehouseCode, "WAR");
    }
    else {
        printf("\nInvalid warehouse code. Please try again\n\n");
        goto reinputwarehouse;
    }
```

```
62      //open warehouse file according to the input warehouseCode
63      warehouseFile = openWarehouseFile(warehouseCode);
64
65      //get record count to create part ID
66      latestRecord = getWarehouseRecordCount(warehouseFile);
67      latestRecord = latestRecord + 1;
68      sprintf(S1.partID, "%s%03d", warehouseCode, latestRecord);
69      printf("\nPart ID\t: %s%03d\n\n", warehouseCode, latestRecord);
70
71      printf("\Enter Section Code: ");
72      gets(S1.sectionCode);
73
74      printf("\nEnter Part Name\t: ");
75      gets(S1.partName);
76
77  reinputquantity:
78      printf("\nEnter Quantity\t: ");
79      scanf("%s", &S1.quantity);
80      getchar();
81
82      //check input quantity whether it is integer
83      if (!isInteger(&S1.quantity)) {
84          printf("\nInput invalid, please try again with number\n");
85          strcpy(S1.quantity, "");
86          goto reinputquantity;
87      }
88
89      S1.linefeed = '\n';
90
91      //add current record to file
92      latestRecord = addWarehouseRecord(warehouseFile, &S1);
93      printf("\n\nRecord successfully created!\n- - - - -\n");
94
95      //get current record count number
96      latestRecord = getWarehouseRecordCount(warehouseFile);
97      printf("Current record count: %d\n", latestRecord);
98
99      //read and print out the latest recorded parts inventory record
100     readWarehouseRecord(warehouseFile, latestRecord, &S1);
101     printInventoryData(&S1);
102
103     //close file
104     closeWarehouseFile(warehouseFile);
105 }
```

partsSupplier.c

This source file enable the user to create a new supplier record and store the record in text file.

```c
#include "partsSupplier.h"
#include "main.h"
#include "fileHandling.h"

//to print supplier data structure
void printSupplierData(partsSupplier* sup)
{
    printf("Supplier ID\t: %s\n", sup->supplierID);
    printf("Supplier Name\t: %s\n", sup->supplierName);
    printf("Location\t: %s\n", sup->location);
    printf("Part ID\t\t: %s\n", sup->partID);
    printf("Part Name\t: %s\n", sup->partName);
}

//main function
void partsSupplierCreation()
{
    //declare pointers, structure and variable
    FILE* supplierFile;
    partsSupplier S2;
    int latestRecord = 0;

    printf("2. Parts Supplier Creation\n- - - - -\n");

    //memory reset partsSupplier structure
    memset(&S2, 0, sizeof(partsSupplier));

    //open supplier file
    supplierFile = openSupplierFile();

    //get supplier data
    printf("*Input existing Supplier ID if there is any\nEnter Supplier ID\t: ");
    gets(S2.supplierID);

    printf("\nEnter Supplier Name\t: ");
    gets(S2.supplierName);
    printf("\nEnter location\t\t: ");
    gets(S2.location);

    printf("\nEnter Part ID\t\t: ");
    gets(S2.partID);

    printf("\nEnter Part Name\t\t: ");
    gets(S2.partName);

    S2.linefeed1 = '\n';

    //call function to add record to file
    latestRecord = addSupplierRecord(supplierFile, &S2);
    printf("\n\nRecord successfully created!\n- - - - -\n");

    //get and print supplier record count
    latestRecord = getSupplierRecordCount(supplierFile);
    printf("Current record count: %d\n", latestRecord);

    //read and print the latest recorded supplier data
    readSupplierRecord(supplierFile, latestRecord, &S2);
    printSupplierData(&S2);

    //close supplier file
    closeSupplierFile(supplierFile);
}
```

partsInventoryUpdate.c

Through this source file, the user can update the quantity of a specific parts inventory record by adding or subtract the value.

```c
#include "partsInventoryCreation.h"
#include "main.h"
#include "fileHandling.h"

//search for the parts that the user wants to update
int partsInventoryFind(FILE* warehouseFile, partsInventory *S1, char *partID)
{
    int totalRecord = getWarehouseRecordCount(warehouseFile);

    memset(S1, 0, sizeof(partsInventory));
    for (int i = 0; i < totalRecord; i++) {
        readWarehouseRecord(warehouseFile, i, S1);
        if (strcmp(partID, S1->partID) == 0) {
            return i;
        }
    }
    return -1;
}

//convert str to int to add quantity
void strAdd(char* strQuantity, int addValue)
{
    int intQuantity;
    intQuantity = atoi(strQuantity);
    //add quantity
    intQuantity = intQuantity + addValue;
    //convert int quantity to string
    sprintf(strQuantity, "%d", intQuantity);
}

//convert str to int to substract quantity
int strSubstract(char* strQuantity, int substractValue)
{
    int intQuantity;
    intQuantity = atoi(strQuantity);
    //if substract value is more than initial quantity, return -1
    if (intQuantity < substractValue) {
        return -1;
    }
    //substract quantity
    intQuantity = intQuantity - substractValue;
    //convert int quantity to string
    sprintf(strQuantity, "%d", intQuantity);
    return 0;
}
```

```
47  //main function
48  void partsInventoryUpdate() {
49      //declare pointers, structure and variable
50      FILE* warehouseFile;
51      partsInventory S1;
52      int quantityUpdate = 0;
53      char selectID[10];
54      char warehouseCode[5];
55      char updateOption[10];
56      int searchResult;
57
58      printf("1. Parts Inventory Update\n- - - - -\n");
59  reinputwarehouse:
60      //memory reset partsInventory structure
61      memset(&S1, 0, sizeof(partsInventory));
62      printf("Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: ");
63      scanf("%s", warehouseCode);
64      getchar();
65
66      //set warehouseCode as the user input
67      if (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) {
68          strcpy(warehouseCode, "WBZ");
69      }
70      else if (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL")) {
71          strcpy(warehouseCode, "WSL");
72      }
73      else if (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR")) {
74          strcpy(warehouseCode, "WAR");
75      }
76      else {
77          printf("\nInvalid warehouse code. Please try again\n");
78          goto reinputwarehouse;
79      }
80
81      //open warehouse file according to the input warehouseCode
82      warehouseFile = openWarehouseFile(warehouseCode);
83
84      //user input part ID to update
85      printf("\nInput part ID to update: ");
86      scanf("%s", &selectID);
87
88      printf("\n\nSearch Result\n- - - - - \n");
89
90      //call function to searchand print the part given part ID's details
91      searchResult = partsInventoryFind(warehouseFile, &S1, selectID);
92      printInventoryData(&S1);
```

```
94      printf("\n1. Add Quantity\n2. Substract Quantity\nSelect Update Option: ");
95      scanf("%s", updateOption);
96      getchar();
97
98  reinputquantity:
99      printf("\nInput Quantity: ");
100     scanf("%d", &quantityUpdate);
101     getchar();
102
103     //add quantity
104     if (0 == strcmp(updateOption, "1") || 0 == strcmp(updateOption, "Add")) {
105         strAdd(S1.quantity, quantityUpdate);
106     }
107     //substract quantity
108     else if (0 == strcmp(updateOption, "2") || 0 == strcmp(updateOption, "Substract")) {
109         if (-1 == strSubstract(S1.quantity, quantityUpdate)) {
110             printf("\nInsufficient quantity to be substracted. Try input the value again");
111             goto reinputquantity;
112         }
113     }
114
115     //update record in file
116     updateWarehouseRecord(warehouseFile, searchResult, &S1);
117     printf("\nUpdated info\n- - - - -\n");
118
119     //print updated record
120     printInventoryData(&S1);
121     //close file
122     closeWarehouseFile(warehouseFile);
123
124 }
```

partsInventoryTracking.c

This source file is created to allow the users to view sorted all the part records' ID in alphabetical order and also view the quantity of parts that is less than ten.

```c
#include "partsInventoryCreation.h"
#include "main.h"
#include "fileHandling.h"

//function to sort parts ID in alphabetical order
int sortPartsAsc(char *warehouseFileName, partsInventory *S1List, int S1ListSize)
{
    //declare pointers, structure and variable
    FILE* warehouseFile;
    partsInventory S1;
    int totalRecord;
    int arraySize = 0;
    int S1ListCount = 0;
    int j;

    //open warehousefile
    warehouseFile = openWarehouseFile(warehouseFileName);
    //get total record count in warehouse file
    totalRecord = getWarehouseRecordCount(warehouseFile);

    //loop line by line to read warehouse record
    for (int i = 0; i < totalRecord; i++) {
        readWarehouseRecord(warehouseFile, i, &S1);

        //add record line to list
        if (S1ListCount == 0)
        {
            memcpy(S1List + S1ListCount, &S1, 64);
        }
        else
        {
            for (j = S1ListCount; j > 0; j--) {
                //data smaller than current array item. move current array item down
                if (0 < strcmp((S1List+ j - 1)->partID, S1.partID)) {
                    memcpy(S1List + j, S1List + j - 1, sizeof(partsInventory));
                }
                else {
                    //copy data to current array item
                    memcpy(S1List + j, &S1, sizeof(partsInventory));
                    break;
                }
            }
            //if j loop until 0 didn't find any smaller item, put data to the first item
            if (j == 0)
                memcpy(S1List, &S1, sizeof(partsInventory));
        }
        S1ListCount += 1;
    }
```

```
49      //close warehouse file
50      closeWarehouseFile(warehouseFile);
51
52      if (S1ListCount > 0)
53          return totalRecord;
54      return 0;
55  }
56
57  //function to get quantity less than 10
58  int quantityLessThan10(char* warehouseFileName, partsInventory* S1List, int S1ListSize)
59  {
60      //declare pointers, structure and variable
61      FILE* warehouseFile;
62      partsInventory S1;
63      int totalRecord;
64      int arraySize = 0;
65      int S1ListCount = 0;
66      int j;
67
68      //memory reset S1list
69      memset(S1List, 0, S1ListSize * sizeof(partsInventory));
70
71      //open warehouse file
72      warehouseFile = openWarehouseFile(warehouseFileName);
73
74      //get total record of the warehouse file
75      totalRecord = getWarehouseRecordCount(warehouseFile);
76
77      //loop to read the warehouse file line by line
78      for (int i = 0; i < totalRecord; i++) {
79          readWarehouseRecord(warehouseFile, i, &S1);
80
81          //convert str quantity to int and check for quantity less than 10
82          if (atoi(S1.quantity) < 10) {
83              //add record to list
84              memcpy(S1List + S1ListCount, &S1, sizeof(partsInventory));
85              S1ListCount++;
86          }
87      }
88
89      //close warehouse file
90      closeWarehouseFile(warehouseFile);
91
92      //return record count
93      if (S1ListCount > 0)
94          return S1ListCount;
95      return 0;
96  }
```

```c
//main function
void partsInventoryTracking()
{
    //declare pointers, structure and variable
    int recordCount;
    char trackingOption[5];
    char warehouseCode[5];
    partsInventory S1List[100];

    //declare list
    char warehouseFile[3][5] = {"WBZ", "WSL", "WAR"};

    printf("Parts Inventory Tracking\n- - - - -\n");
reinputOption:
    printf("\n1. Sort all parts in warehouses\n2. Show stock less than 10 quantity\n\nSelect function, input (1) or (2): ");
    scanf("%s", trackingOption);
    getchar();

    //option 1, sort all parts
    if (0 == strcmp(trackingOption, "1"))
    {
        printf("\n1. Sort all parts in warehouses\n");

        system("cls");
        printf("Sorted Warehouse Data");
        printf("\n- - - - -\n");
        //loop to print record from warehouse by warehouse
        for (int i = 0; i < 3; i++) {
            memset(&S1List, 0, 100 * sizeof(partsInventory));

            //call function to sort all parts from all three warehouses
            recordCount = sortPartsAsc(warehouseFile[i], &S1List[0], 100);

            printf("\n\nWarehouse Code: %s\n", warehouseFile[i]);
            printf("\nRecord count = %d\n", recordCount);
            printf("Part ID\t| Section Code\t| Part Name\t\t| Quantity\t\n");
            printf("-----------------------------------------------------------------\n");
            //loop to print sorted inventory records
            for (int j = 0; j < recordCount; j++) {
                printf("%s\t| ", S1List[j].partID);
                printf("%s\t\t| ", S1List[j].sectionCode);
                printf("%s\t\t| ", S1List[j].partName);
                printf("%s\t\n", S1List[j].quantity);
            }
        }
    }
```

```c
    //option 2, show stock less than 10 quantity
    else if (0 == strcmp(trackingOption, "2")) {
        printf("\n\n2. Show stock less than 10 quantity\n- - - - -\n");
    reinputwarehouse:

        //get warehouse code that the user want to show
        printf("Warehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: ");
        scanf("%s", warehouseCode);
        getchar();

        //set user choice as warehousecode
        if (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) {
            strcpy(warehouseCode, "WBZ");
        }
        else if (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL")) {
            strcpy(warehouseCode, "WSL");
        }
        else if (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR")) {
            strcpy(warehouseCode, "WAR");
        }
        else {
            printf("\n\nInvalid warehouse code. Please try again\n");
            goto reinputwarehouse;
        }

        //memory reset S1list
        memset(&S1List, 0, 100 * sizeof(partsInventory));

        //get record count while call fucntion
        recordCount = quantityLessThan10(warehouseCode, &S1List[0], 100);
        system("cls");
        printf("Stock Quantity that has less then 10 units");
        printf("\n- - - - -\n");

        //print data
        printf("\nWarehouse Code: %s\n", warehouseCode);
        printf("Record Count = %d\n\n", recordCount);
        printf("Part ID\t| Section Code\t| Part Name\t\t| Quantity\t\n");
        printf("-----------------------------------------------------------------------\n");
        //loop to print the filtered record
        for (int j = 0; j < recordCount; j++) {
            printf("%s\t| ", S1List[j].partID);
            printf("%s\t\t| ", S1List[j].sectionCode);
            printf("%s\t\t| ", S1List[j].partName);
            printf("%s\t\n", S1List[j].quantity);
        }
    }
    else {
        printf("\nInput invalid, try again with (1) or (2)\n");
        goto reinputOption;
    }
}
```

searchingFunctionalities.c

This source file is enable the user to search for both parts inventory record and supplier record.

```c
#include "partsInventoryCreation.h"
#include "main.h"
#include "fileHandling.h"
#include "partsSupplier.h"

//function to searh parts inventory
int partsInventorySearch(FILE* warehouseFile, partsInventory* S1, char* partID)
{
    int totalRecord = getWarehouseRecordCount(warehouseFile);

    memset(S1, 0, sizeof(partsInventory));

    //loop to read all record
    for (int i = 0; i < totalRecord; i++) {
        readWarehouseRecord(warehouseFile, i, S1);
        //if record with input part ID found, return
        if (strcmp(partID, S1->partID) == 0) {
            return i;
        }
    }
    return -1;
}

//function to searh parts supplier
int partsSupplierSearch(FILE* supplierFile, partsSupplier* S2, char* partID)
{
    int totalRecord = getSupplierRecordCount(supplierFile);

    memset(S2, 0, sizeof(partsSupplier));

    //loop to read all record
    for (int i = 0; i < totalRecord; i++) {
        readSupplierRecord(supplierFile, i, S2);
        //if record with input part ID found, return
        if (strcmp(partID, S2->partID) == 0) {
            return i;
        }
    }
    return -1;
}
```

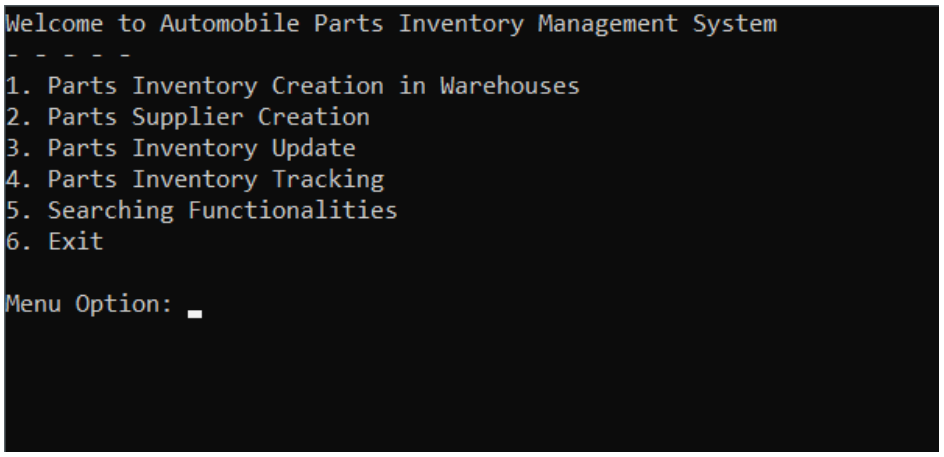
```c
//main function
void searchingFunctionalities() {
    //declare pointers, structure and variable
    FILE* warehouseFile;
    FILE* supplierFile;
    partsInventory S1;
    partsSupplier S2;
    char selectID[10];
    char warehouseCode[5];
    char searchOption[15];
    int searchResult;
    char toMenu[3];

    printf("5. Searching Functionalities\n- - - - -\n");

selectSearchFunction:
    printf("1. Part Record\n2. Supplier Details\nSelect Search Function: ");
    scanf("%s", &searchOption);

    //option 1, search for part record
    if (0 == strcmp(searchOption, "1") || 0 == strcmp(searchOption, "Part Record")) {
reinputwarehouse:
        memset(&S1, 0, sizeof(partsInventory));

        //get warehouse code to search
        printf("\nWarehouse Code\n1. WBZ\n2. WSL\n3. WAR\nSelect Warehouse: ");
        scanf("%s", warehouseCode);
        getchar();

        if (0 == strcmp(warehouseCode, "1") || 0 == strcmp(warehouseCode, "WBZ")) {
            strcpy(warehouseCode, "WBZ");
        }
        else if (0 == strcmp(warehouseCode, "2") || 0 == strcmp(warehouseCode, "WSL")) {
            strcpy(warehouseCode, "WSL");
        }
        else if (0 == strcmp(warehouseCode, "3") || 0 == strcmp(warehouseCode, "WAR")) {
            strcpy(warehouseCode, "WAR");
        }
        else {
            printf("\nInvalid warehouse code. Please try again\n");
            goto reinputwarehouse;
        }

        //open warehouse file with the given warehouseCode
        warehouseFile = openWarehouseFile(warehouseCode);
```

```
87
88              //get part ID to search
89              printf("\nInput Part ID to search: ");
90              scanf("%s", &selectID);
91
92              printf("\n\nSearch Result\n- - - - - \n");
93
94              //call function to search part ID in file
95              searchResult = partsInventorySearch(warehouseFile, &S1, selectID);
96              //print data
97              printInventoryData(&S1);
98              //close file
99              closeWarehouseFile(warehouseFile);
100
101         }
102         //option 2, search for supplier record
103         else if (0 == strcmp(searchOption, "2") || 0 == strcmp(searchOption, "Supplier Details")) {
104             memset(&S2, 0, sizeof(partsSupplier));
105
106             //open supplier file
107             supplierFile = openSupplierFile();
108
109             //get part ID to search
110             printf("\nInput Part ID to search: ");
111             scanf("%s", &selectID);
112
113             printf("\n\nSearch Result\n- - - - - \n");
114
115             //call function to search for supplier data in file
116             partsSupplierSearch(supplierFile, &S2, selectID);
117             //print supplier data
118             printSupplierData(&S2);
119             //close file
120             closeSupplierFile(supplierFile);
121         }
122         else {
123             printf("\nInput invalid, try again with (1) or (2)\n");
124             goto selectSearchFunction;
125         }
126         printf("\n");
127     }
```

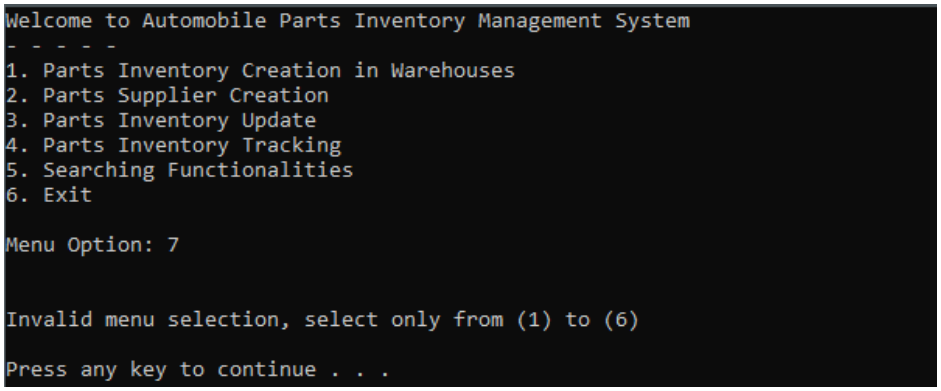
4.0 Program Sample Outputs

4.1 Main Menu

The first output the user received from the program is main menu. The user are able to view the features before selecting it.

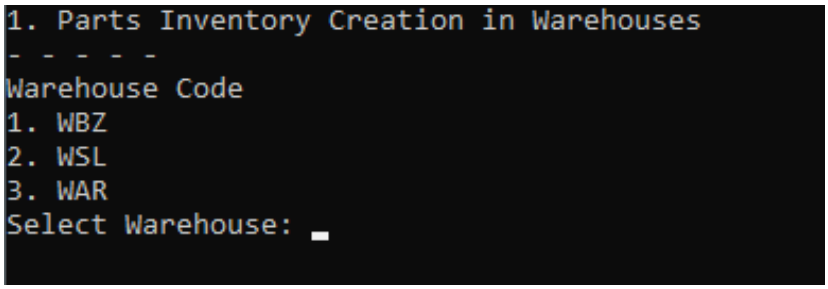


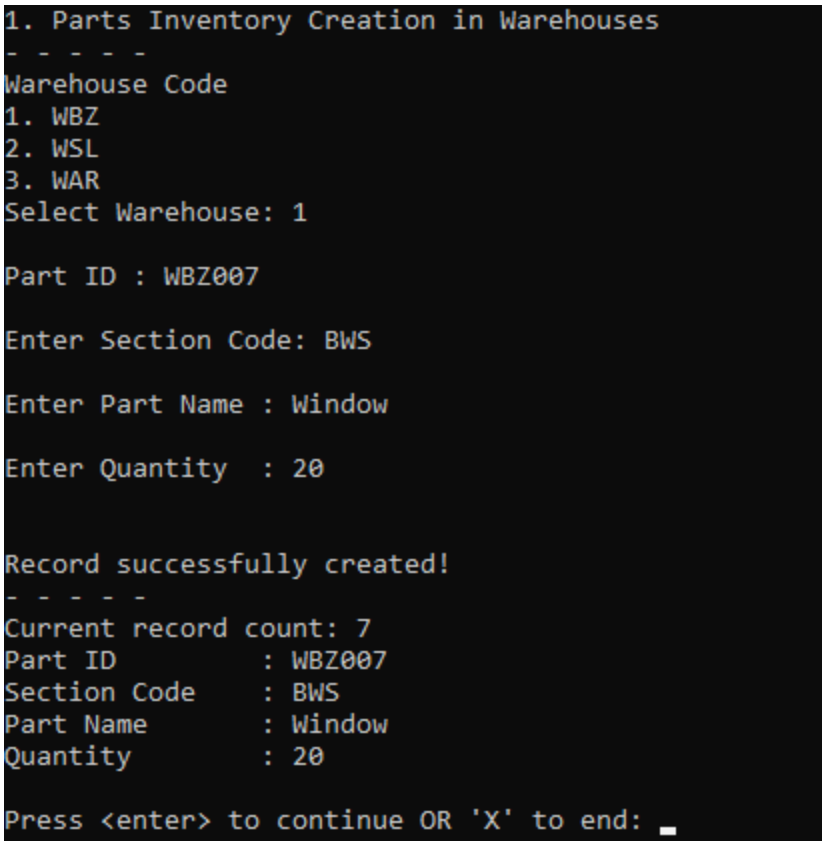
Whenever the user input integer or character that were not found in the menu option, it ill prompt out the instruction once again and allow the user to select.



4.2 Parts Inventory Creation

It was expected the user to first select the warehouse code they wanted to create in. Part ID was automatically generated by a robot. After gathering all the required information to create a parts inventory record, the system will prompt out the summary of the created record.





4.3 Parts Supplier Creation

The user is required to input all the supplier detail. After gathering all the required information to create a parts inventory record, the system will prompt out the summary of the created record.

```
2. Parts Supplier Creation
- - - - -
*Input existing Supplier ID if there is any
Enter Supplier ID       : _
```

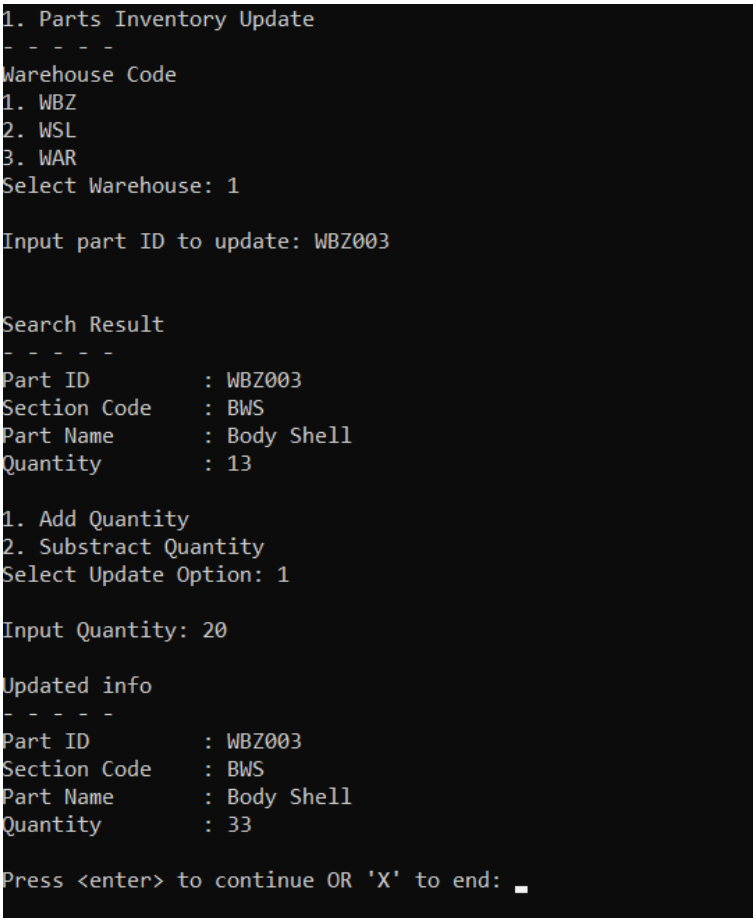
```
2. Parts Supplier Creation
- - - - -
*Input existing Supplier ID if there is any
Enter Supplier ID       : SUP006

Enter Supplier Name     : Sam Inc

Enter location          : Kuala Lumpur

Enter Part ID           : WAR006

Enter Part Name         : Engine


Record successfully created!
- - - - -
Current record count: 19
Supplier ID     : SUP006
Supplier Name   : Sam Inc
Location        : Kuala Lumpur
Part ID         : WAR006
Part Name       : Engine

Press <enter> to continue OR 'X' to end:
```

4.4 Parts Inventory Update

Parts Inventory Update will gather the information that it required to get the part ID, then the user will have to do updates on the quantity of the selected parts record. The system will also prompt out the updated record for the user's references.

```
1. Parts Inventory Update
- - - - -
Warehouse Code
1. WBZ
2. WSL
3. WAR
Select Warehouse:
```

```
1. Parts Inventory Update
- - - - -
Warehouse Code
1. WBZ
2. WSL
3. WAR
Select Warehouse: 1

Input part ID to update: WBZ003


Search Result
- - - - -
Part ID          : WBZ003
Section Code     : BWS
Part Name        : Body Shell
Quantity         : 13

1. Add Quantity
2. Substract Quantity
Select Update Option:
```


1. Parts Inventory Update
- - - - -
Warehouse Code
1. WBZ
2. WSL
3. WAR
Select Warehouse: 1
Input part ID to update: WBZ003
Search Result
- - - - -
Part ID : WBZ003
Section Code : BWS
Part Name : Body Shell
Quantity : 13
1. Add Quantity
2. Substract Quantity
Select Update Option: 1
Input Quantity: 20
Updated info
- - - - -
Part ID : WBZ003
Section Code : BWS
Part Name : Body Shell
Quantity : 33
Press <enter> to continue OR 'X' to end:

4.5 Parts Inventory Tracking

This feature will display all the sorted part record from all three warehouses and also display the stock quantity from a certain warehouse that is less than 10.

```
Parts Inventory Tracking
- - - - -

1. Sort all parts in warehouses
2. Show stock less than 10 quantity

Select function, input (1) or (2): _
```

```
Sorted Warehouse Data
- - - - -


Warehouse Code: WBZ

Record count = 7
Part ID | Section Code  | Part Name            | Quantity
-------------------------------------------------------------------
WBZ001  | ES            | Engine               | 15
WBZ002  | ES            | Radiator             | 10
WBZ003  | BWS           | Body Shell           | 33
WBZ004  | BWS           | Dashboard            | 20
WBZ005  | AS            | Compressor           | 8
WBZ006  | AS            | Cooling Coil         | 2
WBZ007  | BWS           | Window               | 20


Warehouse Code: WSL

Record count = 6
Part ID | Section Code  | Part Name            | Quantity
-------------------------------------------------------------------
WSL001  | BWS           | Body Shell           | 6
WSL002  | ES            | Engine               | 9
WSL003  | BWS           | Dashboard            | 3
WSL004  | BWS           | Bumper               | 7
WSL005  | AS            | Compressor           | 12
WSL006  | ES            | Radiator             | 5


Warehouse Code: WAR

Record count = 6
Part ID | Section Code  | Part Name            | Quantity
-------------------------------------------------------------------
WAR001  | AS            | Compressor           | 15
WAR002  | AS            | Cooling Coil         | 15
WAR003  | ES            | Engine               | 5
WAR004  | BWS           | Body Shell           | 0
WAR005  | BWS           | Side Panel           | 0
WAR006  | BWS           | Bumper               | 5

Press <enter> to continue OR 'X' to end: _
```

```
Parts Inventory Tracking
- - - - -

1. Sort all parts in warehouses
2. Show stock less than 10 quantity

Select function, input (1) or (2): 2


2. Show stock less than 10 quantity
- - - - -
Warehouse Code
1. WBZ
2. WSL
3. WAR
Select Warehouse:
```

```
Stock Quantity that has less then 10 units
- - - - -

Warehouse Code: WBZ
Record Count = 2

Part ID | Section Code  | Part Name            | Quantity
-------------------------------------------------------------------
WBZ005  | AS            | Compressor           | 8
WBZ006  | AS            | Cooling Coil         | 2

Press <enter> to continue OR 'X' to end:
```

4.6 Searching Functionalities

The system prompted out for the user to select whether to search part record or supplier details. In part record, it contains the details such as part ID, section code, part name and quantity. While supplier details contains supplier ID, name, location, supply part ID and supply part name.

```
5. Searching Functionalities
- - - - -
1. Part Record
2. Supplier Details
Select Search Function:
```

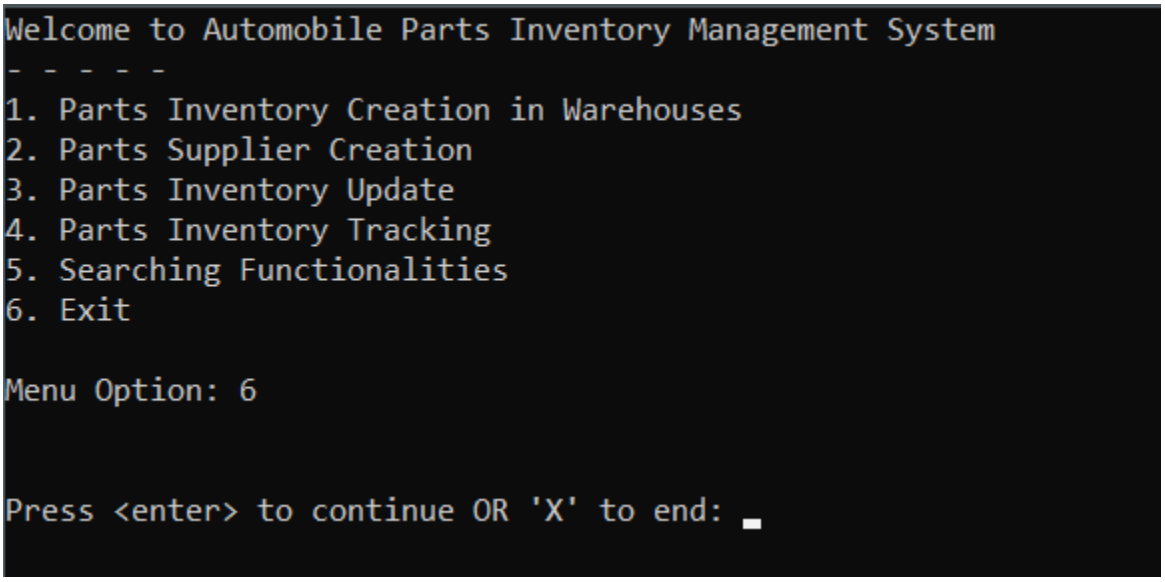
```
5. Searching Functionalities
- - - - -
1. Part Record
2. Supplier Details
Select Search Function: 1

Warehouse Code
1. WBZ
2. WSL
3. WAR
Select Warehouse: 1

Input Part ID to search: WBZ006


Search Result
- - - - -
Part ID         : WBZ006
Section Code    : AS
Part Name       : Cooling Coil
Quantity        : 2

Press any key to continue . . .
```

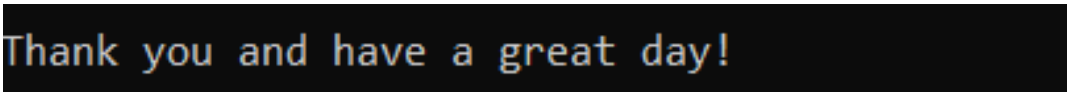
```
5. Searching Functionalities
- - - - -
1. Part Record
2. Supplier Details
Select Search Function: 2

Input Part ID to search:
```

```
5. Searching Functionalities
- - - - -
1. Part Record
2. Supplier Details
Select Search Function: 2

Input Part ID to search: WAR003


Search Result
- - - - -
Supplier ID     : SUP001
Supplier Name   : Saiko Manufacture
Location        : Selangor
Part ID         : WAR003
Part Name       : Engine

Press any key to continue . . .
```

4.7 Exit

To exit the program, the system displays the menu option allowing the user to quit the program. After input ‘6’, the program will once again print out a confirmation sentence to allow the user to quit the program. Before exiting the program, the system prompted out a thank you sentence to the user.





5.0 Text Files

5.1 Supplier File - Supplier.txt

All the supplier information such as supplier ID, supplier name, supplier location, part ID and part name are stored under “Supplier.txt”.

```
SUP001    Saiko Manufacture   Selangor        WBZ001    Engine
SUP001    Saiko Manufacture   Selangor        WBZ002    Radiator
SUP001    Saiko Manufacture   Selangor        WSL002    Engine
SUP001    Saiko Manufacture   Selangor        WAR003    Engine
SUP002    TMG Auto            Johor           WBZ003    Body Shell
SUP002    TMG Auto            Johor           WBZ004    Dashboard
SUP003    Lear Corp           Kuala Lumpur    WSL001    Body Shell
SUP003    Lear Corp           Kuala Lumpur    WSL003    Dashboard
SUP003    Lear Corp           Kuala Lumpur    WSL004    Bumper
SUP004    Remco Auto          Johor           WAR001    Compressor
SUP004    Remco Auto          Johor           WAR002    Cooling Coil
SUP002    TMG Auto            Johor           WSL005    Compressor
SUP005    NYT Automotive      Kedah           WAR004    Body Shell
SUP005    NYT Automotive      Kedah           WAR005    Side Panel
SUP001    Saiko Manufacture   Selangor        WBZ005    Compressor
SUP001    Saiko Manufacture   Selangor        WBZ006    Cooling Coil
```

5.2 Warehouse Inventory File

Parts inventories are recorded under three files individually according to their warehouse code, namely, “WBZinventory.txt”, “WSLinventory.txt” and “WARinventory.txt”. The files record the part details such as part ID, Section code, part name and quantity of the part.

5.2.1 WBZinventory.txt

```
WBZ001    ES   Engine          15
WBZ002    ES   Radiator        10
WBZ003    BWS  Body Shell      13
WBZ004    BWS  Dashboard       20
WBZ005    AS   Compressor      8
WBZ006    AS   Cooling Coil    2
```

5.2.2 WSL inventory.txt

```
WSL001    BWS  Body Shell      6
WSL002    ES   Engine          11
WSL003    BWS  Dashboard       3
WSL004    BWS  Bumper          7
WSL005    AS   Compressor      12
```

5.2.3 WARinventory.txt

```
WAR001    AS   Compressor      15
WAR002    AS   Cooling Coil    15
WAR003    ES   Engine          5
WAR004    BWS  Body Shell      0
WAR005    BWS  Side Panel      0
```

6.0 Conclusion

In conclusion, the car parts inventory management system encompasses the entire process of automobile parts management, including inventory generation in warehouses, inventory updates, inventory tracking, and searching capabilities. I learned basic programming skills through this project, including how to plan a programme using flowcharts and pseudocode, as well as how to construct a system with working features using the C programming language. Conditioning, structure, bubble sort and more are also utilised in the programme to only acquire reliable information about the parts to continue the programme in order to produce a system with fewer mistakes.