COM2002 INTERMEDIATE PROGRAMMING
2024 – 2025 SPRING

Laboratory Week: 10 – 14 March 2025

Topic : Structures, Unions, and Enumerations

Program : Inventory (inventory.c)

Definition : The program tracks parts stored in a warehouse. Information about the parts is stored in an array of structures. Contents of each structure: Part number, Name, Quantity.

Operations supported by the program:

- Add a new part number, part name, and initial quantity on hand. Prints an error message and returns prematurely if the part already exists or the database is full.
- Given a part number, print the name of the part and the current quantity on hand if the part number is found; otherwise prints an error message.
- Given a part number, change the quantity on hand if the part already exists; otherwise print an error message.
- Print a table showing all information in the database.
- Terminate program execution.

The program prompts the user to enter an operation code, then calls a function to perform the requested action. The codes i (insert), s (search), u (update), p (print), and q (quit) will be used to represent these operations. Repeats until the user enters the command 'q'. Prints an error message if the user enters an illegal code.

Modify *read_line* function in Laboratory 4:

The version of *read_line* in *Chapter 13:Strings* won't work properly in the current program. Consider what happens when the user inserts a part:

Enter part number: 528

Enter part name: Disk drive

The user presses the Enter key after entering the part number, leaving an invisible new-line character that the program must read. When *scanf* reads the part number, it consumes the 5, 2, and 8, but leaves the new-line character unread. If we try to read the part name using the original *read_line* function, it will encounter the new-line character immediately and stop reading. This problem is common when numerical input is followed by character input. One solution is to write a version of *read_line* that skips white-space characters before it begins storing characters. This solves the new-line problem and also allows us to avoid storing blanks that precede the part name.

Expected output:

```
Enter operation code: i
Enter part number: 528
Enter part name: Disk drive
Enter quantity on hand: 10

Enter operation code: s
Enter part number: 528
Part name: Disk drive
Quantity on hand: 10

Enter operation code: s
Enter part number: 914
Part not found.

Enter operation code: i
Enter part number: 914
Enter part name: Printer cable
Enter quantity on hand: 5

Enter operation code: u
Enter part number: 528
Enter change in quantity on hand: -2

Enter operation code: u
Enter part number: 915
Part not found.

Enter operation code: u
Enter part number: 914
Enter change in quantity on hand: +5

Enter operation code: p
Part Number   Part Name                 Quantity on Hand
    528       Disk drive                       8
    914       Printer cable                   10

Enter operation code: w
Illegal code

Enter operation code: i
Enter part number: 25
Enter part name: Printer
Enter quantity on hand: 4

Enter operation code: p
Part Number   Part Name                 Quantity on Hand
     25       Printer                          4
    528       Disk drive                       8
    914       Printer cable                   10

Enter operation code: q
```