CSE 1142 - COMPUTER PROGRAMMING II
Programming Assignment # 3

DUE DATE: 10/05/2018 - 23:59 (No extension)

In this homework you will simulate a simple cargo system for a firm.

1. In this system, there exists a list of forbidden items that cannot be transported. Create a two dimensional character array *forbiddenItems* to store the forbidden items.

- Size of the first dimension of the array will be a constant value which is defined as a macro. Therefore, if the value of the constant changes, then the size of the first dimension changes. You can write 6 as the initial value of the constant.
- You can assume that item names do not exceed 50 characters.
- If an item name consists of more than one word, these words are combined together with underscore "_" sign. For example, if the name of the item is "cutlery pack" it will be stored as "cutlery_pack" in the array.

2. Customers of the firm have unique customer numbers and the firm also keeps how many cargo transactions are done by each customer. To store the customer numbers and their transaction counts, create a two dimensional integer array *customers*.

- The first dimension of *customers* array stores the customers. The firm can serve to at most 500 customers.
- The second dimension of the array should store the total number of transaction counts.
- Customer numbers should be greater than zero.
- For each valid transaction, total transaction count of the corresponding customer is incremented.

3. If a cargo is damaged, the firm should cover the loss. For this reason, the firm wants the customers to put a price on their goods before delivery. Create a two dimensional double array *transactions* to store approximate price of each transaction of the customers.

- Each customer can make at most 730 transactions.
- Each row of this array belongs to a unique customer and row number of a customer is the same as the row number that his/her customer number is stored in the *customers* array.

4. When you start the system a welcome message will appear on the screen and the system administrator starts to enter forbidden items.

```
Hello Admin Please Enter the Forbidden Items:
mirror
cutlery_pack
knife
parfume
medicine
computer
```

a) Store the items entered by the system administrator into the *forbiddenItems* array.

b) Print “Do you want to start the system? 1 --> Yes, 0 --> No:” message and if the answer is equal to 0 end the program immediately.

c) If the answer is equal to 1, then a user starts to write the transactions of the customers.

```
Hello Admin Please Enter the Forbidden Items:
mirror
cutlery_pack
knife
parfume
medicine
computer
Do you want to start the system? 1 --> Yes, 0 --> No:
1
Welcome to Cargo Tracking System
Please Enter the Customer Number :
```

d) The user first writes the customer number. If a customer with the entered number does not exist in the customers array you should add the customer. If there exists a customer with the entered number you will not add again. To understand whether a customer exists or not in the customer array you will write **searchCustomer** function :

- **int searchCustomer(int customerNumber)** : function searches the customers array and detects whether a customer with the given *customerNumber* exists or not and returns array index of the customer if finds in the array. Otherwise, the function should return -1.

5. After the first part, the user starts to enter items contained in the cargo.

```
Hello Admin Please Enter the Forbidden Items:
mirror
cutlery_pack
knife
parfume
medicine
computer
Do you want to start the system? 1 --> Yes, 0 --> No:
1
Welcome to Cargo Tracking System
Please Enter the Customer Number :
100
Please Enter the Content of the Cargo :2 bag 5 book 1 cutlery_pack 3 notebook 2 mirror
enter the approximate value for one of 3 notebook
```

- For each item, number of it and its name is written. Items are separated from each other by using at least 1 space between the previous item name and number of the next item. For example, if 2 bags and 5 books are to be sent then the entered string can be written as :
  - `2 bag 5 book`
- You can assume that the list of items to be read does not exceed 500 characters.
- You should read items as a whole line (represented as a single string). In other words, you cannot read items partially, or using formatting.

- There can be many extra spaces in the string and there is no limit for the number of extra spaces.
  - Please Enter the Content of the Cargo :`2 bag 5 book 1 cutlery_pack 3 notebook 4 mirror`
    - For example, the entered string above, does not contain any space in the beginning of the string (before 2). There is just 1 space between the item counts and item names and there is no space at the end of the string (after mirror).
  - Please Enter the Content of the Cargo **:** `               2               bag 5            book               1 cutlery_pack                  3 notebook                 4                         mirror     `
    - For example, the entered string above, contains more than one space in the beginning of the string (before 2), between item counts and item names (i.e between 2 and bag), between previous item name and next item count (i.e between bag and 5) and at the end of the string (after mirror).

6. After reading the content of the cargo, you will call **getContent** recursive function.

- **void getContent(char content[], double *money)**
- *char content[]* : content of the cargo that you have read.
- *double *money*: calculated approximate value of the cargo. This value will be used in main function to print the price of the cargo content.
- This function stops when the content is empty or contains just spaces.
  - The function should invoke **checkFinish** function firstly to check whether the content is empty or contains just spaces and call this function inside **getContent** function.
    - **int checkFinish(char stringToCheck[])**
    - *char stringToCheck[]* : character array to be evaluated for being empty or containing just spaces.
- Then, you should remove forbidden items if there exists any in the content of the cargo.
  - To remove forbidden items, write **removeForbiddens** recursive function and call it inside **getContent** function.
    - **void removeForbiddens(char content[], int itemIndex)**
    - *char content[]* : content of the cargo

- *int itemIndex* : index of last element of the *forbiddenItems* array. You should change the *itemIndex* for each recursive call.

- This function checks whether any of the forbidden items is contained in the content of the cargo or not. It starts checking from the last element of the *forbiddenItems* array and stops when the first element is checked. If it finds any forbidden item, that item and its count is removed from the content string. **Since this function is a recursive function and calls for itself you cannot write for/while loop(s) to iterate through all elements of the *forbiddenItems* array.**

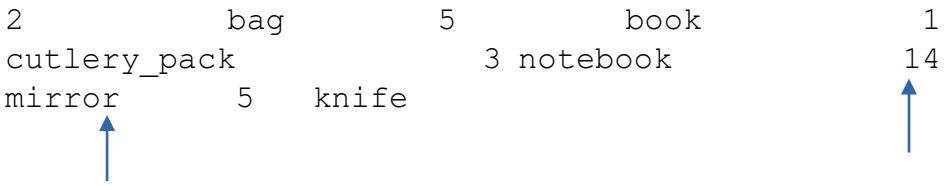
- A specific forbidden item can exist in the content more than one time and repetitive occurrences should also be removed from the content. **Since this function is a recursive function and calls for itself you cannot write for/while loop(s) to check repetitive occurrences of forbidden items.**

- To check the occurrences of each forbidden item in an order, you should call the **searchWord** function. **searchWord** function searches for occurrence of a string in another string.

  - **int searchWord(char str1[], char str2[])**

    - occurrence of string *str2* in string *str1* is checked.

    - this function determines the start and end index of *str2* in *str1* if it exists in *str1*. If *str2* = mirror, start and end indexes are shown below for *str1* (given below)

      - 

        end = index of r start = index of 1

    - **searchWord** function returns 1 if *str2* occurs in *str1*, returns 0 otherwise.

    - **You should write your own function and you cannot use built-in functions to compare strings.**

    - **You should deal with extra spaces; therefore, you cannot use built-in functions and formatters to separate a string into words.**

- If **searchWord** function finds a forbidden item, calls **reconstruct** function to remove this item from the string. Start and end parameters of the **reconstruct** function will be the start and end indices that you have found in **searchWord** function.
  - **void reconstruct(char stringToConstruct[], int start,int end)**
  - **reconstruct** function updates the *stringToConstruct* char array by extracting the substring between *start* and *end* indices. You should keep the extracted substring in *stringToSeach* char array.

- For example, if string below (a) is entered as the content of the cargo, first forbidden items are removed from the string. 12 knife is removed from the string firstly, since the search is done starting from last element of the *forbiddenItems* array (b). Secondly, 5 knife is removed because knife occurs for the second time (c). Thirdly, 1 cutlery_pack is removed from the string (d). Finally, 14 mirror is deleted from the string (e). The text is painted with blue color and spaces are printed as dots to make spaces visible to you.

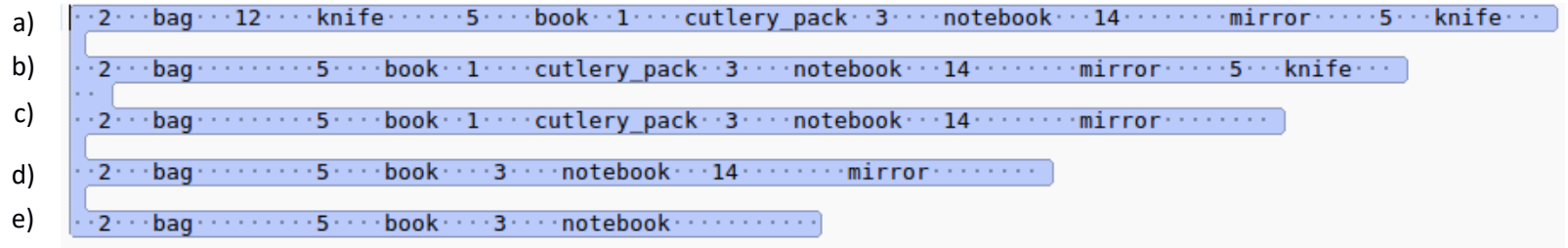


Figure 1

7. After removing forbidden items, remaining items are separated from the string. For this purpose, **seperateItem** function is called in **getContent** function to detect and separate each item from the string.

- **void seperateItem(char wholeString[], int *itemCount, char item[])**
- *char wholeString[]* : is the string that contains items.
- *int *itemCount*: number of the separated item will be stored in this address.
- *char item[]:* is the item to be detected and separated from the string.
- For example, if the string in Fig. 1e is the first argument of the **separateItem** function, "2 bag" is stored in *item[]* array and 2 is stored in address pointed by *itemCount* pointer.
- This function calls **reconstruct** function to separate "2 bag" from the *wholeString*.
- Since **getContent** function is a recursive function you will not call **separateItem** function in while/for loop(s).

8. For each separated item you will request approximate price for it in **getContent** function and calculate the cumulative sum of them

- For example, after obtaining string in Fig.1e, the program wants approximate value for notebook, book and bag items :

```
Please Enter the Customer Number :
100
Please Enter the Content of the Cargo :  2   bag   12    knife     5    book  1    cutlery_pack  3    notebook   14       mirror     5   knife
enter the approximate value for one of 3    notebook
1.2
enter the approximate value for one of 5    book
10
enter the approximate value for one of 2   bag
2.2
price 58.000000
Please Enter the Customer Number :
```

- Here price is calculated as (1.2 * 3) + (10 * 5) + (2 * 2.2) = 58.0 in the **getContent** function and printed to the screen in **main** function.
- If the user enters an empty string or a string that contains just spaces than the price should be calculated as 0.
- If the item count of an item is less than or equal to zero price will not be calculated for that item and a warning message should be printed on the screen.

```
Please Enter the Customer Number :
100
Please Enter the Content of the Cargo :-5 bag  10 book
enter the approximate value for one of 10 book
2.5
Price for item -5 bag  will not be calculated because of having invalid number of items
price 25.000000
Please Enter the Customer Number :
```

9. In **main** function, you will read customer numbers and content of the cargo until a negative value is entered for a customer number.

- If a customer number is entered for the first time, you will add this customer as a row of the 2 dimensional *customers* array. The column of the *customers* array should keep the total number of transactions belonging to customers. If there exists a valid transaction for a customer, you will increment the number of valid transactions stored in the column of corresponding customer row.

- Calculated price of valid transactions will be stored in the *transactions* array in the same row as the *customers* array starting from the first empty column.
- If a customer number is entered before, you will not add this customer again to the *customers* array. If the new transaction is a valid one, then increment the transaction count of the customer. Additionally, you should add the calculated price for the second transaction to the corresponding row of the *transactions* array.

10. Invalid transactions:

- If an empty string or a string which contains just spaces is entered for the content of the cargo, then the transaction count of the customer should not be incremented. However, if this customer number is entered for the first time you will add this customer number to customers array.
- If content of a cargo consists of just forbidden items; then, this is an invalid transaction and will not be added to the *transactions* array. However, if this customer number is entered for the first time you will add this customer number to customers array.
- If the user enters 0 as the price of the all items in the cargo, then this transaction will not be added to the *transactions* array because the price is equal to 0. However, if this customer number is entered for the first time you will add this customer number to *customers* array.
- If all item counts are less than or equal to 0, then this transaction will not be added to the *transactions* array because the price is equal to 0. However, if this customer number is entered for the first time you will add this customer number to *customers* array.
- If not all but some of the item counts are less than or equal to 0 and user enters 0 as the price of all remaining items, then this transaction will not be added to the *transactions* array because the price is equal to 0. However, if this customer number is entered for the first time you will add this customer number to *customers* array.

11. When a negative number is entered for the customer number, call **printCustomers** function to print all customer numbers and their total transaction count and price of each transaction.

- **void printCustomers()**

12. Sample output:

```
Hello Admin Please Enter the Forbidden Items:
mirror
cutlery_pack
knife
parfume
medicine
computer
Do you want to start the system? 1 --> Yes, 0 --> No:
1
Welcome to Cargo Tracking System
Please Enter the Customer Number :
100
Please Enter the Content of the Cargo :  2   bag   12    knife      5    book  1    cutlery_pack  3    notebook   14        mirror     5   knife
enter the approximate value for one of 3    notebook
5.5
enter the approximate value for one of 5    book
2.5
enter the approximate value for one of 2   bag
10.25
price 49.500000
Please Enter the Customer Number :
102
Please Enter the Content of the Cargo :
price 0.000000
Please Enter the Customer Number :
101
Please Enter the Content of the Cargo :8  pillow
enter the approximate value for one of 8  pillow
10
price 80.000000
Please Enter the Customer Number :
103
Please Enter the Content of the Cargo :   5  computer  7   knife
price 0.000000
Please Enter the Customer Number :
102
Please Enter the Content of the Cargo :5  computer   -9   chair     1  table
enter the approximate value for one of 1  table
100.25
Price for item -9   chair   will not be calculated because of having invalid number of items
price 100.250000
Please Enter the Customer Number :
104
Please Enter the Content of the Cargo :8  chair   2  table
enter the approximate value for one of 2  table
0
enter the approximate value for one of 8  chair
0
price 0.000000
```

```
Please Enter the Customer Number :
100
Please Enter the Content of the Cargo :    2  television    1     table
enter the approximate value for one of 1     table
100.5
enter the approximate value for one of 2  television
2000
price 4100.500000
Please Enter the Customer Number :
104
Please Enter the Content of the Cargo :0  chair    1    table
enter the approximate value for one of 1    table
0
Price for item 0  chair  will not be calculated because of having invalid number of items
price 0.000000
Please Enter the Customer Number :
-10
```

```
User 100 has sent 2 cargos price of each is as follows
price for cargo 1 -> 49.500000  price for cargo 2 -> 4100.500000

User 102 has sent 1 cargos price of each is as follows
price for cargo 1 -> 100.250000

User 101 has sent 1 cargos price of each is as follows
price for cargo 1 -> 80.000000

User 103 has sent 0 cargos price of each is as follows


User 104 has sent 0 cargos price of each is as follows


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

This is a simple scenario to test your implementation. There might be other test cases too. Therefore, please pay attention to use the function and variable names in your implementations. You cannot decrease the number of functions.

Submission Instructions

Please zip and submit your files using filename YourNumberHW3.zip
(ex: 150713852HW3.zip) to Canvas system (under Assignments tab).
Your zip file should contain the following files:

1. C source file: cargo_yourStudentNumber.c (ex: cargo_150713852.c)

Notes:

1. Write a comment at the beginning of each program to explain the purpose of the program.
2. Write your name and student ID as a comment.
3. Include necessary comments to explain your actions.
4. Select meaningful names for your variables.
5. You are allowed to use the materials that you have learned in lectures & labs.
6. Do not use things that you did not learn in the course.
7. In case of any form of **copying and cheating** on solutions, all parts will get **FF** grade. You should submit your own work. In case of any forms of cheating or copying, both giver and receiver are equally culpable and suffer equal penalties.
   All types of plagiarism will result in FF grade from the course.
8. No late submission will be accepted.

Grading:

- checkFinish function → 5 points
- reconstruct function → 5 points
- searchWord function → 12 points
- removeForbiddens function → 12 points
- separateItem function → 10 points
- getContent function → 15 points
- searchCustomer function → 5 points
- printCustomer function → 5 points
- main function → 6 points
- The correctness of test cases → 20 points
- Submission Format → 5 points
  - C source file
  - Make sure that your source file can be compiled and run on another computer.
  - Make sure that the input/output of your program must be the same with the examples above (all informative strings & spaces). Otherwise, auto-grader cannot grade your code!!!
  - Comments are necessary!