COP 3223 Spr 2019 – Section 1
Introduction to C - Programming Assignment #3
Section 1 Due date: See syllabus

Objectives

1. To reinforce your knowledge of 'for' loops and 'while' loops
2. To utilize strings (character arrays) in C
3. To implement logic in a program that is menu-based and is the seed for a complex, practical software system.

Problem: Food Bank Program

A Food Bank is a non-profit organization which distributes food items to people in need during local emergency food programs. People come in to a Food Bank and make their donations, and others might come in to make a request for any food item they want.

A Food Bank Program is used to keep track of the stock the Food Bank has (i.e., the donations), keep a record of requests that come in, fulfilling requests and printing the status report which consists of the amount of stock present in the food bank and the requests that are not yet fulfilled.

The program would require you to make two tables: Donations Table and Request Table. Each table will have two attributes: the Inventory Type, which is what type of food item it is, e.g., Milk, Meat, etc., and the Amount, which is the amount of that food item, e.g., 20, 30, etc. Note: For simplification, no units for amounts are considered.

Functionality Specification

1. Add donations and requests to the tables.
2. For any new donation that comes in, check if any donation is present in the donations table with same inventory type; if there is one then increase its amount by the new donation amount, and do not add a new entry in the donations table. If not, then add a new entry in the donations table for that donation.
3. Remove a request from the request table when it is fulfilled.
4. If for a request the request amount is greater than the amount present, then partially fulfill that request by the amount present. And do not delete that request from the request table; just reduce its amount by the amount that is fulfilled.
5. If in the process of fulfilling the request, the amount present for any donation becomes zero, then remove that donation from the donations table.
6. Print the status report, containing stock currently available and requests that are not yet fulfilled.
7. When the option is to Fulfill a Request, you are to only process the one request that is at the top of the Request Table (the oldest request). Thus, if it cannot be fulfilled, it will stall the system, until someone donates enough, so it can be satisfied later.

Input/Output Specification

At the beginning, the program should present the user with the available options, and ask for his/her preference. Based on the preference, the program should perform the corresponding task.

The user will have five choices:

1. Enter a Donation
2. Enter a Request
3. Fulfill the earliest Request
4. Print status report
5. Exit

When the user selects ‘1’, the program should ask the user for the inventory type and amount of the donation to be made. When the donation has been added to the donations table, the program should notify the user by printing out “Donation Added”.

When the user selects ‘2’, the program should ask the user for the inventory type and amount of the request to be made. When the request has been added to the requests table, the program should notify the user by printing out “Request Added”.

When the user selects ‘3’, the program should print “Request Fulfilled” if the request is completely fulfilled, print “Cannot be Fulfilled” if there is no donation of that type, and print “Partially Fulfilled” if the donation type exists, but cannot completely satisfy the request. If a request is fulfilled, it should be deleted from the Requests Table; if a donation is fully used up in satisfying a request, the Donation Table entry should be deleted.

When the user selects ‘4’, the program should print the current Donations Table and the Requests Table.

When the user selects ‘5’, the program should print “Thank You for running the software. Bye for now”, and then the program exits.

Implementation Hints

If we assume that the maximum number of donated types at any time is 100, and that the Maximum length of a description of the type is 19, then, our tables can be declared as:

```
char donations_inv_type[100][20];
int donations_amount[100];
```

You will need to have variables to keep track (count) of the total donated types, and total requests, in the two tables.

Use function ‘strcmp’ to compare if two strings are equal. Remember that when the two strings are equal this function returns 0.

Use function ‘strcpy’ to copy from one string variable to another.

While removing the request from the requests table once it is fulfilled, you will have to shift up the requests that are below that request in the request table, to take the place of the removed request. The same applies when you remove a donation, if the donation amount becomes zero.

For every new donation you will have to check if the inventory type of that donation is present in the donations table or not. For this, use the 'strcmp' function.
For a new request that comes in, you do NOT have to perform the check for whether any request is present in the request table with the same inventory type. Just add a new line for each new request in the requests table, as every request corresponds to a different family.

Assume that all inputs are perfectly correct, so you don't have to do any error checking on the input.

Output Sample

```
Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_1_

Enter inventory type: MILK
Enter the amount: 20

Donation Added!
Press any key to continue . . .

Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_2_

Enter inventory type: MILK
Enter the amount: 5

Request Added!
Press any key to continue . . .

Welcome to the Food Bank Program
```

```
   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_1_

Enter inventory type: FRUIT
Enter the amount: 10

Donation Added!
Press any key to continue . . .

Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_1_

Enter inventory type: MILK
Enter the amount: 5

Donation Added!
Press any key to continue . . .

Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_4_

Printing the Donations Table

MILK 25
FRUIT 10

Printing the Requests Table
```

```
MILK 5

Press any key to continue . . .

Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_3_

-------- Fulfilling Requests--------

Request Fulfilled

Press any key to continue . . .

Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit

Enter your choice:_4_

Printing the Donations Table

MILK 20
FRUIT 10

Printing the Requests Table


Press any key to continue . . .

Welcome to the Food Bank Program

   1. Add a donation
   2. Add a request
   3. Fulfill a request
   4. Print status report
   5. Exit
```

```
Enter your choice:_5_

Thank You for using the software. Bye for now.

Press any key to continue . . .
```

References

Notes: Arrays, 2D Arrays, Strings

Deliverables

One source file: *foodbank.c*, for your solution to the given problem.

All files are to be submitted over WebCourses/Canvas. **(Do NOT submit .cpp files!!!)**

Restrictions

Although you may use other compilers, your program must compile and run using Dev C++. Please use Dev C++ to develop your program. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details

Your program will be graded upon the following criteria:
1. Your correctness.
2. Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 5% deducted from your grade.
3. Whether or not you have designed reasonable functions to solve the task at hand.
4. Compatibility with Dev C++ (in Windows). If your program does not compile in this environment, you will get a **sizable** deduction from your grade.