Assignment #6

Introduction to C Programming – COP 3223

Objectives

1. To learn how to use arrays to store and retrieve data
2. To reinforce knowledge of previous programming constructs

Introduction: Programmers for a Better Tomorrow

Programmers for a Better Tomorrow is an organization dedicated to helping charities, medical societies, and scholarship organizations manage various tasks so that they can focus on making the world a better place! They have asked you and your classmates to help them develop some new programs to benefit their organizations.

Problem: Food Bank Management (foodbank.c)

Programmers for a Better Tomorrow is now in partnership with a local Food Bank which distributes food items to people in need. 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 Management 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 will require you to make at least one array. You may choose to make multiple one-dimensional arrays: one to serve as a Donations Table and one to serve as a Request Table. The index of the array will represent the inventory type. The value in the array will represent the amount present or needed of that type.

You may, instead, choose to combine these to two arrays into a single two-dimensional array.

There are 5 inventory types:

0. Protein
1. Dairy
2. Grains
3. Vegetables
4. Fruits

Your program should allow the user the following options:

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

Option 1:
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".

Option 2:
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".

Option 3:
Check each type of inventory in the Request Table.

For each item that has a value in the request table, check to see if there is inventory of the same type in the donations table.

If there are no donations of that type, print "<type x> requests cannot be fulfilled." Replace <type x> with the correct inventory type (Dairy, Fruit, etc.).

If there is some inventory in the donations table, but not enough to process all requests, print "<type x> requests will be partially fulfilled". Reduce both the donation and request amounts based on how much of the request was able to be fulfilled. Replace <type x> with the correct inventory type (Dairy, Fruit, etc.).

If there is enough inventory to fulfill the request, print "type x> requests will be fulfilled." Reduce the appropriate amount from the Donations Table and remove the amount from the Request Table. Replace <type x> with the correct inventory type (Dairy, Fruit, etc.).

Option 4:
Print both tables, indicating which is for donations and which is for requests.

After options 1, 2, 3, and 4 prompt the user with the menu again.

Option 5:
Print "Thank you for running our system!" and then the program exits.

Do not prompt the user for any more information.

<u>Sample Data Items</u>
You may wish to use these data structures to help you solve the problem.

```
int status[2][5];
char TYPES[5][20] = {"PROTEIN", "DAIRY", "GRAINS", "VEGETABLES", "FRUITS"};
```

The values in "TYPES" are strings and may be printed with %s. For example, to print "DAIRY" we could use printf("%s", TYPES[1]);

<u>**Input Specification**</u>

1. All inputs will be greater than or equal to zero

<u>**Output Specification**</u>

View the sample run to see how the output tables should be formatted.

<u>**Output Sample**</u>

Below is a sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above.

In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

<u>**Sample Run**</u>

```
Welcome to the Food Bank Management Program!
What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
1

What donation type would you like to enter?
        0. Protein
        1. Dairy
        2. Grains
        3. Vegetables
        4. Fruits
0

How many would you like to enter? 5
Donation Added.

What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
1

What donation type would you like to enter?
        0. Protein
        1. Dairy
```

```
            2. Grains
            3. Vegetables
            4. Fruits
2

How many would you like to enter? 10
Donation Added.

What would you like to do?
            1. Enter a Donation
            2. Enter  a Request
            3. Fulfill Requests
            4. Print status report
            5. Exit
1

What donation type would you like to enter?
            0. Protein
            1. Dairy
            2. Grains
            3. Vegetables
            4. Fruits
4

How many would you like to enter? 7
Donation Added.

What would you like to do?
            1. Enter a Donation
            2. Enter  a Request
            3. Fulfill Requests
            4. Print status report
            5. Exit
4
            Protein:        Donations: 5     Requests: 0
            Dairy:          Donations: 0     Requests: 0
            Grains:         Donations: 10    Requests: 0
            Vegetables:     Donations: 0     Requests: 0
            Fruits:         Donations: 7     Requests: 0

What would you like to do?
            1. Enter a Donation
            2. Enter  a Request
            3. Fulfill Requests
            4. Print status report
            5. Exit
1
```

```
What donation type would you like to enter?
        0. Protein
        1. Dairy
        2. Grains
        3. Vegetables
        4. Fruits
0

How many would you like to enter? 3
Donation Added.

What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
2

What would you like to request?
        0. Protein
        1. Dairy
        2. Grains
        3. Vegetables
        4. Fruits
1

How many would you like to request? 5
Request Added.

What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
2

What would you like to request?
        0. Protein
        1. Dairy
        2. Grains
        3. Vegetables
        4. Fruits
2

How many would you like to request? 15
Request Added.
```

```
What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
4
        Protein:        Donations: 8    Requests: 0
        Dairy:          Donations: 0    Requests: 5
        Grains:         Donations: 10   Requests: 15
        Vegetables:     Donations: 0    Requests: 0
        Fruits:         Donations: 7    Requests: 0

What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
3

Dairy requests cannot be fulfilled.
Grain requests will be partially fulfilled.

What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
4
        Protein:        Donations: 8    Requests: 0
        Dairy:          Donations: 0    Requests: 5
        Grains:         Donations: 0    Requests: 5
        Vegetables:     Donations: 0    Requests: 0
        Fruits:         Donations: 7    Requests: 0

What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
6

Sorry, that was not a valid input.
```

```
What would you like to do?
        1. Enter a Donation
        2. Enter  a Request
        3. Fulfill Requests
        4. Print status report
        5. Exit
5
Thank you for running our system!
```

Deliverables
One source file – *foodbank.c* – is to be submitted over WebCourses.

Restrictions
Although you may use other compilers, your program must compile and run using Code::Blocks. 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 programs 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 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.