Who gets the chocolates?

REVISED: THIS ASSIGNMENT HAS BEEN REVISED FROM THE EARLIER VERSION POSTED ON MIMIR. YOU WILL NO LONGER USE ASSERT TO CHECK THE THE NUMBER OF COMMAND LINE ARGUMENTS NOR TO CHECK IF THE INPUT FILE OPENED APPROPRIATELY.

IF THE COMMAND LINE ARGUMENTS ARE INCORRECT PRINT THE FOLLOWING STATEMENT.
Not enough command line arguments! Exiting the program!

IF THE FILE DID NOT OPEN SUCCESSFULLY PRINT THE FOLLOWING STATEMENT.
File did not open successfully! Exiting the program!

If you choose this assignment you could earn up to 105/100.

For this assignment, you will recreate the following scenario:

N students are sitting around a table. They are numbered 1 through N. There is a bowl of chocolates in the center of the table, and each student around the table takes a single piece of chocolate, one at a time in ascending order, starting with someone specified in the input. You will need to determine which student does not get to choose a chocolate from the bowl.

The input file will have 3 numbers. The first number is the number of students at the table. The second number is the number of chocolates in the bowl. The third number represents the student that will take the first chocolate from the bowl. The first name of each student sitting at the table is also in the input.

For instance, if your input is:

5 2 3
Yvon
Toby
Russell
Katie
Teila

Your output should be:

The following student(s) did not receive chocolate! Maybe next time!!
Yvon
Toby
Teila

To explain the output:

The 5 indicates there are 5 students at the table. The 2 says there are 2 chocolates in the bowl on the table. The 3 indicates that Russell, the 3rd name in the list, gets to choose a chocolate from the bowl first, then Katie. Since there are only two chocolates in the bowl, Yvon, Toby, and Teila did not get a piece of chocolate.

I will provide the functions.h file. You will need to add the header guards and the needed include statements.

You will also need to provide the appropriate comments above each function. The comments should follow the format described below:

Return: (if there is a return you are to explain what is being returned. If no return then put void.)

Parameters: (list the type of parameters and a short description of what it will be used for in the function.)

Description: (You are to write a detailed description of the functions purpose and how it will accomplish its purpose. In other words, describe what it is doing. It does not matter if the function is small and self-explanatory, you must provide a description.)

A substantial number of points will be deducted if you do not provide appropriate documentation.

Below is a description of each function needed for this assignment. You will implement these functions in the functions.c file.

char readData(FILE*, int*, int*, int*);**
The FILE is the file pointer for the input file.
The int*'s represent the number of students, the number of chocolates in the bowl, and the number that represents the student that will be the first to choose a chocolate from the bowl. This function will read the three integers, then the names of the students. The names will be stored in a 2d character array. Because you do not know the number of students at compile time, you **must** dynamically allocate a 2d character array for the student names. (THIS IS NOT AN OPTION. I AM REQUIRING DYNAMIC ALLOCATION.)

Hint: You cannot simply set a character array equal to another character array. You should use strcpy.

void Winner(int, int, int, char);**
This is where you will determine which students chose chocolates and which did not.

void printStudents(int , char);**
This function prints the actual output.

void freeMemory(char, int);**
This function will free the dynamically allocated memory.

You **ARE** allowed to make minor changes to the function prototypes. However, these changes must be minor changes. You are still required to implement these functions. You may add functions if you wish.

driver.c

Driver.c should have a minimal amount of code.
Suggested steps for main.

SEE ABOVE INSTRUCTIONS

Create and open a file pointer. ~~You should use the "C" assert function to check if the file pointer is NULL. If you are not familiar with assert, look it up. You should also use assert to determine if the appropriate number of command line arguments are used when running the program.~~
Call readData.
Call Winner.
Call printStudents.
Close the file pointer.
Call freeMemory.