CS 260
Spring 2015
Programming Assignment 3

- Write a Python3 program project3.py that operates as described below.
- Your program will read from standard input and write to standard output. The input has four lines. The first three lines are integer values (n, f, b) and the fourth line is a string (s). The output will be a single integer value, which is explained below.
- Your program will implement a doubly-linked circular list. *Do not use Python's built-in list, or array, or any other pre-defined type!*
- Begin by constructing a linked list with values 0,1,2,3,…,n. Initially the current position is at node 0, and the current direction is forward.
- The size of the list can be very large, so your program should use efficient linked list operations. Otherwise your program might be too slow for some larger data sizes.
- Your linked list must support at least these six operations:

Toggle the current direction between forward and backward
Move either f nodes forward or b nodes backward, depending on the current direction
Increment the current node's value, modulo n+1, so the value remains in range 0…n
Decrement the current node's value, modulo n+1, so the value remains in range 0…n
Copy the current node's value to a new node; insert this new node *before* the current node if current direction is forward, or *after* the current node if current direction is backward
Remove the current node; print the removed value if the list becomes empty; otherwise move to the *previous* node if current direction is forward, or to the *next* node if current direction is backward

- The input string s indicates a sequence of list operations to be performed. Each character of s must be the first letter of one of the six linked list operations described above (either uppercase or lowercase). Your program will perform the indicated sequence of list operations, and continue repeating this same sequence of operations as many times as necessary, until the list becomes empty and the final removed value is displayed.
- Example: Here are an input file input0.txt and its corresponding output file output0.txt, which can be created via this command:
  python3 project3.py < input0.txt > output0.txt

5 2 1 mrimrtmcd	4

- Here is an annotated trace using this input file input0.txt:

	Initial constructed list	-> 0 <-> 1 <-> 2 <-> 3 <-> 4 <-> 5 <->
M	Move forward f=2 nodes	-> 2 <-> 3 <-> 4 <-> 5 <-> 0 <-> 1 <->
R	Remove node 2	-> 1 <-> 3 <-> 4 <-> 5 <-> 0 <->
I	Increment node 1 to 2	-> 2 <-> 3 <-> 4 <-> 5 <-> 0 <->
M	Move forward f=2 nodes	-> 4 <-> 5 <-> 0 <-> 2 <-> 3 <->
R	Remove node 4	-> 3 <-> 5 <-> 0 <-> 2 <->
T	Toggle direction to backward	-> 3 <-> 5 <-> 0 <-> 2 <->
M	Move backward b=1 node	-> 2 <-> 3 <-> 5 <-> 0 <->
C	Copy node 2	-> 2 <-> 2 <-> 3 <-> 5 <-> 0 <->
D	Decrement node 2 to 1	-> 1 <-> 2 <-> 3 <-> 5 <-> 0 <->
M	Move backward b=1 node	-> 0 <-> 1 <-> 2 <-> 3 <-> 5 <->
R	Remove node 0	-> 1 <-> 2 <-> 3 <-> 5 <->
I	Increment node 1 to 2	-> 2 <-> 2 <-> 3 <-> 5 <->
M	Move backward b=1 node	-> 5 <-> 2 <-> 2 <-> 3 <->
R	Remove node 5	-> 2 <-> 2 <-> 3 <->
T	Toggle direction to forward	-> 2 <-> 2 <-> 3 <->
M	Move forward f=2 nodes	-> 3 <-> 2 <-> 2 <->
C	Copy node 3	-> 3 <-> 2 <-> 2 <-> 3 <->
D	Decrement node 3 to 2	-> 2 <-> 2 <-> 2 <-> 3 <->
M	Move forward f=2 nodes	-> 2 <-> 3 <-> 2 <-> 2 <->
R	Remove node 2	-> 2 <-> 3 <-> 2 <->
I	Increment node 2 to 3	-> 3 <-> 3 <-> 2 <->
M	Move forward f=2 nodes	-> 2 <-> 3 <-> 3 <->
R	Remove node 2	-> 3 <-> 3 <->
T	Toggle direction to backward	-> 3 <-> 3 <->
M	Move backward b=1 node	-> 3 <-> 3 <->
C	Copy node 3	-> 3 <-> 3 <-> 3 <->
D	Decrement node 3 to 2	-> 2 <-> 3 <-> 3 <->
M	Move backward b=1 node	-> 3 <-> 2 <-> 3 <->
R	Remove node 3	-> 2 <-> 3 <->
I	Increment node 2 to 3	-> 3 <-> 3 <->
M	Move backward b=1 node	-> 3 <-> 3 <->
R	Remove node 3	-> 3 <->
T	Toggle direction to forward	-> 3 <->
M	Move forward f=2 nodes	-> 3 <->
C	Copy node 3	-> 3 <-> 3 <->
D	Decrement node 3 to 2	-> 2 <-> 3 <->
M	Move forward f=2 nodes	-> 2 <-> 3 <->
R	Remove node 2	-> 3 <->
I	Increment node 3 to 4	-> 4 <->
M	Move forward f=2 nodes	-> 4 <->
R	Remove node 4	4

The following rules apply to all projects in this course:

- We now provide a virtual machine so you can test your program in the same environment where it will be graded. This server is "`cs260.ua.edu`". Regardless where you primarily develop your program, you must use this server to test your projects before submitting. Login using your Bama userid and password. You might first need to install the SSH secure shell and file transfer clients, for example at http://helpdesk.ua.edu/~recovery/ssh.htm. Also, if you wish to access the server from off-campus, you'll need to install a VPN client, which is available at https://mybama.ua.edu on the Tech tab.
- During grading, your program will be tested on the `cs260.ua.edu` server. Example scripts and input/output files will be provided on this server in the /projects directory.
- We provide a common platform that all students must use to test their programs before submitting, and we will use this same platform to evaluate each submitted program. Your program will NOT be evaluated based on how it performs in any other environment, because we will not attempt to replicate different environments used by each individual student.
- In previous semesters, some students did not follow the instructions and/or did not adequately test their programs before submitting, so they lost points for errors that they could have easily detected and corrected. This semester, it is required to test the final version of your program on the server before you submit it. All the scripts and input/output files are provided on the server in the /projects directory. When you run the script, it will produce a file "checksums.txt" that you must submit along with your program. When grading your project, we will use the contents of this file to verify that you actually did test the final version of your program on the server.
- You are permitted to verbally discuss ideas with classmates or other people, but sharing any code is strictly forbidden. **Do not look at your classmates' or anybody else's code, and do not show anybody your code!** However, you are encouraged to verbally discuss the projects with your classmates, provided nobody involved is looking at any code during the discussion.
- You must write all your own code for each project. Do not take code from any other source. If you happen to find a similar program online or in a book, you are NOT permitted to copy portions of that program into your program. The purpose of the projects is to learn how to write programs for data structures and algorithms, not to practice googling for solutions.
- **If your code is too similar to another student's code or to anybody else's code, you will receive an invitation to discuss the situation in the Dean's office.** We use the MOSS system which detects plagiarism based on programs that are too similar. The penalty for plagiarism is typically a score of 0 on the project or a grade of F in the course.
- You should strive to make each program as efficient as possible. During testing, we will allow each program to run for a reasonable amount of time, but if your program takes much longer to run than we think it should, we will eventually decide your program is too inefficient (or possibly in an infinite loop), and in this case we may terminate the execution of your program before it ends.

- Place all your program's source files and the file "checksums.txt" (generated by the script) into a single zip file, give it a name of the form LastName_FirstName_ProjectNumber.zip (example: Doe_John_1.zip), and upload this zip file using Blackboard. Make sure your zip file includes the latest version of all your source files. Your zip file should contain the file "checksums.txt" and all the source files needed to run your program, and these files should all be located at the top level of your zip file. The zip file should not contain any other subfolders. Do not submit projects via email.
- Each assignment will be due by 11:59pm on the due date specified on Blackboard. Please aim to submit your project earlier than the due date in case you encounter unforeseen circumstances at the last minute. There will be a substantial deduction for late submissions. Points will also be deducted if you do not follow all the instructions exactly, because this can make it difficult or impossible to grade your project.
- After the projects are graded, your score will be uploaded onto Blackboard, along with a grading sheet that summarizes how your score was calculated. Also, any additional input/output files used when grading the projects will be placed in the /projects directory on the `cs260.ua.edu` server.
- To completely understand your project's score, you will likely need to run your program on the server using the new input/output files. If you think your project grade was computed incorrectly, you must inform us **within at most one week** from when your score is posted on Blackboard. You must stop by one of the instructors' offices during office hours and show us that your program runs correctly using the script and input/output files located on the server. If a mistake was made when grading your program, we will be glad to correct it.
- However, note that the following statements (which we hear quite frequently) will NOT cause us to increase your project score:
    - "I spent many hours working on the project, so I want a higher score."
    - "My program contains over 1000 lines of code, so I want a higher score."
    - "I didn't follow all the instructions because I thought they were optional."
    - "My program shouldn't be tested on any valid inputs that I forgot to consider."
    - "I didn't understand the project description, and I didn't ask for clarification."