COSC2430 Homework 3: Lists, Stacks, Postfix, and Prefix

1. **Introduction**
   You will create a C++ program to update their linked list and stack so that the nodes can be passed back and forward between a list and a stack.The purpose of this homework is to get students familiar with the Lists and stacks.
2. **Input, command, and output**
   a. Input file
      The input file contains a list of postfix/prefix equations where each equation needs to be added to the **end** of your **Doubly linked list (needs head,tail,next, & prev)**

      postfix: A B+ CD-*
      prefix:+ *A BC

      1. The first word will be either **postfix** or **prefix**
      2. This word will always be followed a colon **:**
      3. This will be followed by that type of equation
         a. The equations will be made up of:
            i. upper and lower case letters
            ii. Operations: +-*/
            iii. Random spaces **(these will need to be removed)**

   b. command file
      This contains a large list of commands. Commands followed by (postfix/prefix/position/all) will contain one of those strings in parentheses this tells you which nodes it will be affecting **(add an extra endl to every print statement if it passes test3.sh then you implemented the endl correctly)**

      - (postfix): this modifies all postfix nodes
      - (prefix): this modifies all prefix nodes
      - (position): will contain an int ex: (3)
      - (all): affects every single node

      Reserve Stack: (this stack needs to be manually implemented for full credit)
      In addition to your doubly linked list you will need a reserve stack the command file will tell you when to add/remove nodes from the stack or how to modify the stack

      1. **convertList (postfix/prefix/position/all)**
         a. Converts everything to the other type of equation, that fits the parameter in the parentheses if it give all it will convert every node in the list from postfix to prefix and every prefix to postfix
            i. If position is <= 0 modify the first value
            ii. If position is >= size of list do nothing
      2. **removeList (postfix/prefix/position/all)**
         a. Removes and deletes all nodes that fit the parameter.
            i. If position is <= 0 delete first node
            ii. If position is >= size do nothing
      3. **printList**

a. Prints the list in order starting with the word List: (print EMPTY if its empty)
    i. EX:
       List:
       postfix:AB+CD-*
       prefix:+*ABC
    ii. EX: (if list is empty)
       List:
       EMPTY

4. **printListBackwards**
    a. Prints the list in reverse order (prints EMPTY if list is empty)
       Ex:
       List:
       prefix:+*ABC
       postfix:AB+CD-*
5. **pushReserve (postfix/prefix/position/all)**
    a. removes all nodes that fit description from list and adds to top of reserve stack **(if multiple nodes fit the description then you add the front nodes first)**
        i. If position is <= 0 remove the first node
        ii. If position is >= size do nothing
6. **popReserve (position)**
    a. adds node to certain position in list from reserve stack
        i. if pos <= 0 add to front
        ii. if pos >= size add to end
7. **flipReserve**
    a. reverses the Reserve Stack
8. **printReserveSize**
    a. prints size of the reserve
       Ex:
       Reserve Size: 3
9. **convertReserve**
    a. Converts the top of the reserve stack. If postfix changes to prefix and vice versa.
10. **printReserveTop**
    a. Prints the top member of the reserve stack
       EX:
       Top of Reserve: postfix:AB+CD-*
       EX: (if empty)
       Top of Reserve: EMPTY
11. **emptyReserve (position)**
    a. pops off each node from the stack until the stack is empty adding all members to a specific location

i. if pos <= 0 add to front
ii. if pos >= size add to end

c. output file
   i. The output file contains everything that the command file prompts it to print (if the command file doesn't contain any print statements then the output file will be empty)
d. Examples
   Linux Command:
   ./stack "input=input31.txt;command=command31.txt;output=output31.txt"
   i. Example 1
      Input31.txt

      postfix: A B+ CD-*
      prefix:+ *A BC

      Command31.txt

      printList
      printListBackwards
      convertList (postfix)
      printList
      convertList (prefix)
      printList
      convertList (all)
      printList
      convertList (0)
      printList

      Output31.txt

      List:
      postfix:AB+CD-*
      prefix:+*ABC

      Reversed List:
      prefix:+*ABC
      postfix:AB+CD-*

      List:
      prefix:*+AB-CD
      prefix:+*ABC

List:
postfix:AB+CD-*
postfix:AB*C+

List:
prefix:*+AB-CD
prefix:+*ABC

List:
postfix:AB+CD-*
prefix:+*ABC

ii. Example 2
Input32.txt

postfix: BAC-*
prefix:+A*B/C-EF
postfix:FE-C/B*A+
postfix:AB-C-D/
postfix:AB-CF*-D / E+

command 32.txt

printList
pushReserve (postfix)
printList
printReserveSize
printReserveTop
convertReserve
flipReserve
popReserve (0)
printList
emptyReserve (0)
printList

Output32.txt

List:
postfix:BAC-*
prefix:+A*B/C-EF
postfix:FE-C/B*A+
postfix:AB-C-D/
postfix:AB-CF*-D/E+

List:
prefix:+A*B/C-EF

Reserve Size: 4

Top of Reserve: postfix:AB-CF*-D/E+

List:
postfix:BAC-*
prefix:+A*B/C-EF

List:
prefix:+/--AB*CFDE
postfix:AB-C-D/
postfix:FE-C/B*A+
postfix:BAC-*
prefix:+A*B/C-EF

iii. Example 3
Input33.txt

prefix: +*X / B C / F G
postfix: B X + T G * H /- A+
postfix: A L D M U P*****
prefix:--------Q E D FGHVCS
postfix:LED+-SCM*+-G/
prefix:+++++QWERTY

Command33.txt

printList
pushReserve (all)
printList
emptyReserve (999)
printList

output33.txt

List:
prefix:+*X/BC/FG
postfix:BX+TG*H/-A+
postfix:ALDMUP*****

prefix:--------QEDFGHVCS
postfix:LED+-SCM*+-G/
prefix:+++++QWERTY

List:
EMPTY

List:
prefix:+++++QWERTY
postfix:LED+-SCM*+-G/
prefix:--------QEDFGHVCS
postfix:ALDMUP*****
postfix:BX+TG*H/-A+
prefix:+*X/BC/FG

3. **Requirements**

Homework is individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. Code that is copied from another student (for instance, renaming variables, changing for and while loops, changing indentation, etc, will be treated as copy) will be detected and result in ”0” in this homework. The limit is 50% similarity.

4. **Turn in your homework**

Homework 1 needs to be turned in to our Linux server, follow the link here https://rizk.netlify.app/courses/cosc2430/2_resources/
Make sure to create a folder under your root directory, name it hw1 (name need to be
lower case), only copy your code to this folder, no testcase or other files needed.
PS: This document may have typos, if you think something illogical, please email TAs for
confirmation.