ID2230 HW-1
Deadline: Sep 22, 2024

1. The goal of this exercise is to:

   (a) Create a stack ADT using arrays, with a maximum size of 500 and each element in the stack being of type double.

   (b) Use your stack ADT to evaluate a given postfix expression.

   Input Format: The input consists of a single line that contains a sequence of tokens (each token being an integer or binary operator in +,-,*,/) separated by commas, and ending in the character '#'.

   Example: 5, 3, 4, -, -2, *, /, #

   Output: A single number with a precision of 3 digits after the decimal point. If the given expression is not a postfix expression, output Invalid Expression. The output in the above input example will be 2.500.

   NOTE: A template myStack.h and a template main program are shared. You must write a corresponding myStack.c program, and write out the entire main program. For the main program, you may deviate from the template.

2. Create a **circular linked list**, in which every node has a next and previous pointer, and no node is distinguished as head or tail; a pointer to any one node in the list will be used to access the list. Each node will have two attributes: a userID and a name. An example is shown below; the numbers in the nodes are for illustration only.

   Your circular linked list should support the following operations:

   (a) insert(x,y), which inserts a node y as a successor to x, and returns a pointer to y.

   (b) delete(x), which deletes node x and returns a pointer to its successor.

   (c) printList(x), which prints all the key values of the elements of the circular list containing x in cyclic order, starting with x.

   (d) merge(x,y): Given two pointers x, y to elements of two lists L_1 and L_2 respectively, merge the two lists and return a pointer to an element in the combined list.

   For the full list of operations, see below.

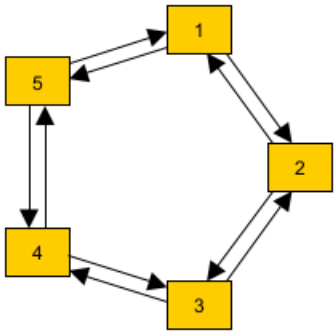

Figure 1: A circular linked list

NOTE: A template circularList.h file and a main program are shared. You must write the corresponding circularList.c file so that the main program works correctly. It is necessary to implement all the operations in the header file except for the ones marked Optional. Note that other sequences of insert/delete/merge may also be tested, so you are encouraged to generate some test cases of your own.