T-201-GSKI, GAGNASKIPAN

Vor 2018

Recursion

Assignment grading. For full marks your solution needs to be accepted by Mooshak. Points may be deducted for

- not implementing the solution according to the specifications,
- redundant or repeated code,
- messy code,
- code that is difficult to understand and not explained in comments.

If your solution is not accepted by Mooshak, it will receive a maximum grade of 7.

PREFIX EXPRESSION PARSER 50%

You are to implement a prefix expression parser using recursion. A prefix parser is a program which takes input in the form of prefix notation, also called polish notation, i.e. / 10 5 and returns the outcome which is 2. Prefix notation doesn't need any parenthesis do always be unambiguous, for example:
+ 5 - + 3 4 2 = (+ 5 (- (+ 3 4) 2)) = 5 + ((3 + 4) - 2) = 10
Another example of prefix notation is - * / 15 - 7 + 1 1 3 + 2 + 1 1 = 5.

THE IMPLEMENTATION

Implement the function parseExpression which takes an input stream as a parameter and returns an integer. The function should read the input from the stream, one token at a time (valid tokens being '+', '-', '*', '/' and any integer), and evaluate the calculations recursively. The function should be able to compute the value of any expression written in polish notation.
If the program would ever have to divide by zero it should instead throw a DivideByZeroException.

EXTRA

For more info about prefix notation see: https://en.wikipedia.org/wiki/Polish_notation

REVERSE LIST 50%

You are given the class *ListNode* which implements a singly linked list. The data in each node is a single character (*char data*).
A list is simply the series of nodes linked by the ***next*** link on the nodes, and the list *ends* as soon as the next link on a node is *NULL*.
You are also given the class *ListOperations* which has several static functions that operate on lists, taking the first node as an argument and in some cases returning the first node of a new list. In all cases it is ***not*** necessary to make new nodes, only to change the ***next*** links, and you are expected to implement this functionality with recursion.
You will ***not*** be working with a list class that stores a head and tail and other information. Your only entry point into the list is the first node itself. The first node in a singly-linked list ***is*** the list itself. When the operation returns it also returns the first node in the list, whether or not that is still the same node.

Your assignment is to implement the operation **ListNode* reverseList(ListNode* head)**

- ***head*** is the first node in linked list. Turn all the next links around so that the list is reversed and return the new first node.

Endurkvæmni

PREFIX EXPRESSION PARSER 50%

Þú átt að útfæra prefix expression parser með endurkvæmni. Prefix parser er forrit sem tekur inntak á formi sem kallað er prefix notation eða polish notation og lítur svona út: / 10 5. Þessi segð myndi skila 2.
Prefix notation þarf enga sviga en er samt alltaf laus við tvíræðni, t.d.:
+ 5 - + 3 4 2 = (+ 5 (- (+ 3 4) 2)) = 5 + ((3 + 4) - 2) = 10
Annað dæmi um prefix notation er - * / 15 - 7 + 1 1 3 + 2 + 1 1 = 5.

ÚTFÆRSLA

Útfærið fallið parseExpression sem tekur inn inntaksstraum og skilar heiltölu. Fallið á að lesa inntak frá straumnum, einn tóka í einu (gildir tókar eru '+', '-', '*', '/' og allar heiltölur), og framkvæma útreikningana með endurkvæmni. Fallið á að geta reiknað segðir sem skrifaðar eru í polish notation. Ef að forritið stendur frammi fyrir því að þurfa að deila með núlli skal það þess í stað kasta DivideByZeroException.

EXTRA

Frekari upplýsingar um prefix notation: https://en.wikipedia.org/wiki/Polish_notation

REVERSE LIST 50%

Gefinn er klasinn *ListNode* sem útfærir eintengdan lista. Gögnin í hverjum hnút eru stakur stafur (*char data*).
Listi er einfaldlega röð hnúta, tengdra á next bendi hnútsins, og listinn *endar* þar sem ***next*** bendir hnúts hefur gildið *NULL*.
Einnig er gefinn klasinn *ListOperations* sem inniheldur nokkur static föll sem vinna með lista. Þau taka öll inn fyrsta hnútinn í lista og skila í sumum tilfellum fyrsta hnút nýs lista. Í öllum tilfellum er ***óþarfi*** að búa til neina nýja hnúta heldur þarf bara að breyta ***next*** bendunum. Ykkur er ætlar að útfæra lausnina með endurkvæmni.
Þið munuð ***ekki*** vinna með klasa sem inniheldur head og tail og aðrar upplýsingar. Eina tengingin við listann er gegnum fyrsta hnútinn sjálfan. Fyrsti hnúturinn í eintengdum lista ***er*** listinn sjálfur. Þegar fallið skilar skilar það einnig fyrsta hnútnum, hvort sem hann er ennþá sami hnúturinn eða nýr hnútur.

Verkefnið er að útfæra aðgerðina **ListNode* reverseList(ListNode* head).**

- ***head*** er fyrsti hnúturinn í tengdum lista. Snúið öllum next bendunum við þannig að listinn sé í öfugri röð og skilið nýja fremsta hnútnum.