Linked Lists And Stacks

January 20, 2009

1 Problem #1

1.1 Description

A number of operations that can be done on a x list are defined below. The input will have several operations. Perform these operations using a linked list and output as expected.

Input	Action
P	Print the linked list.
N	Print the number of elements in the list.
S	Sort the linked list in non decreasing order by manipulating pointers.
E X	insert X at the end of list
B X	insert X in the beginning of the list
D X	Delete the first occurrance of the node with element X. (Your list need not be sorted)
R	Reverse the list with the help of a recursive function.
S X	Search for an element X in the list. Print TRUE if it exists and FALSE if the number is not present in the list. (Your list need not be sorted)
DL	Delete the complete list and free the memory.

1.2 Input Format:

N followed by N instructions one on each line.

1.3 Output Format:

Output corresponding to each operation.

1.4 Sample Input

```
8
E 3
B 1
E 10
S
```

```
P
N
R
P
```

1.5 Sample Output:

```
1 3 10
3
10 3 1
```

2 Problem #2

2.1 Description

A digital computer in AD 4th century used linked lists as its internal representation of positive integers (in the binary form). Read two positive integers, ione and itwo (could be very long integers), and carry out one of the following arithmetic operations to get the third (ithree). The input numbers are given in the decimal form.

S x	Print the number of nodes in the linked list representation of x (1 2 or 3)
A	Add ione and itwo and store result in ithree.
M	Multiply ione and itwo and store result in ithree.
P	Print ithree in decimal form.

2.2 Input Format:

A sigle line consisting of two integers *ione* and *itwo*. And then N followed by N instructions one in each line.

2.3 Output Format:

As described above.

2.4 Sample Input

```
9347538975384535 328749237423423947
6
S 1
S 2
M
S 3
A
P
```

2.5 Sample Output:

```
54
59
112
338096776398808482
```

3 Problem #3

3.1 Description

Read two strings from the keyboard character by character. Each string is terminated by a new line. Represent both of them as linked lists in the memory with start pointing to the first characters. Now remove from the first string all characters which are present in the second string.

3.2 Input Format:

First string on $line_1$ and second on $line_2$

3.3 Output Format:

Single line

3.4 Sample Input

```
RAILROAD CROSSING WITHOUT ANY VOWELS
AEIOUY
```

3.5 Sample Output:

```
RLRD CRSSNG WTHT N VWLS
```

4 Problem #4

4.1 Description

A 2D matrix is represented in a linked manner. Every element in the structure has two links. One to an element on its right and another to an element below it. When there is no further elements, a *NULL* is inserted. Dimension of the matrix is not explicitly stored. Matrix is represented by the pointer to the first element.

You are given two matrices. Add them. Return NULL, if the dimensions do not match for addition. If matrices are compatible for addition, return the sum-matrix. Prototype of the function is given below.

```
matrix * addMat(matrix *a, matrix *b);
```

You are given two matrices. M_1 : given by m, n followed by $m * n$ elements and M_2 : given by p,q followed by $p * q$ elements. Print the result of the sum of the matrices row-wise.

4.2 Input Format:

First line will have m and n and the next m rows each having n integers correspond to $matrix_1$. Then a line with p and q and after that the next p rows each having q integers correspond to $matrix_2$.

4.3 Output Format:

As described.

4.4 Sample Input

```
2 3
2 3 4
4 5 6
2 3
6 7 8
1 2 3
```

4.5 Sample Output:

```
8 10 12
5 7 9
```

5 Problem #5

In this problem you are given a mathematical expression either in infix or prefix or postfix format. You have to write a program to print it in all three forms and also it's value. For the sake of uniqueness when you output in infix format remove unnecessary parenthesis. For example if the output is *((1))* you must print *1*.

5.1 Input

Each case spans a line. You can assume that the values in the expression are only integers. Input ends with EOF. There is a space after every entity in the expression. (to keep it simple).

5.2 Ouput

For each case output in the following manner. Notice that each output spans only 4 lines.

```
prefix
infix
posfix
value
```

5.3 Sample Input:

```
( ( 1 ) )
1 2 +
( ( 1 ) + ( 2 ) ) * ( 3 )
( 1 * 2 ) + 3
```

5.4 Sample Output

```
1
1
1
1
+ 1 2
1 + 2
1 2 +
3
* + 1 2 3
( 1 + 2 ) * 3
1 2 + 3 *
9
+ * 1 2 3
1 * 2 + 3
1 2 * 3 +
5
```