IC250 Laboratory Assignment - 04

Aditya Nigam

August 31, 2016

IC250 Programming and Data Structure Practicum

Lab Assignment No. 04; Date : {6,7,8} Sep , 2016

Problems

1. You have to code any one out of these two questions :

   - Write a C code that can detect that given string is a Palindrome or not using Stack.

     ```
     $./pal_stack
     $ Please enter the string (or type 0 to exit) :
     $ sachin
     $ String "sachin" is NOT a Palindrome.
     $ Please enter the string (or type 0 to exit) :
     $ malayalam
     $ String "malayalam" is a Palindrome
     ```

   - Write a C code that can detect that the left and right parenthesis are balanced or not in any input expression using Stack. For example expression : $((a + s))$ balanced; $((a - n)$ unbalanced.

     ```
     $./par_stack
     $ Please enter the expression (or type 0 to exit) :
     $ ((a+b))
     $ Expression "((a+b))" is having balanced parenthesis.
     ```

2. You have to code any one out of these two questions :

   - Write down a C function that can perform stack reversal using recursion.
   - Write down a C function that can reverse any linked list.

3. Write down a C function that can implement a Stack using a Queue data structure (supporting standard operations like enqueue() and dequeue()). One method is by making *push* costly and other can be by making *pop* costly. Code any one.

   ```
   Method 1 (By making push operation costly) :

   This method makes sure that newly entered element is always
   at the front of q1, so that pop operation just dequeues from
   q1, q2 is used to put every new element at front of q1.

   Push(s, x) // x is the element to be pushed and s is stack
    1) Enqueue x to q2
    2) One by one dequeue everything from q1 and enqueue to q2.
    3) Swap the names of q1 and q2
   ```

```
    // Swapping of names is done to avoid one more movement
    of all elements from q2 to q1.

pop(s)
    1) Dequeue an item from q1 and return it.
```

4. Write a C function to transpose any sparse matrix efficiently ($Fast\_Transpose$) as discussed in the lecture.