# Homework06 – Singly Linked List

**Due 11:55 PM Saturday, April 19, 2014, 3 + 3 points, individual assignment**

## Purpose of Assignment

- The goal of this homework is to get familiar with the linked list data structure.

## Assignment 1 (3 points)

1. Complete the singly linked list or SList.cpp given including the following items;
   - void freeSList(pSList p) // deallocate all the nodes in the list and itself
   - int deleteSList(int item, pSList p); // delete a node with item, return true or false
   - int deleteFrontSList(pSList p); // delete a node at front, return true or false
   - int deleteEndSList(pSList p); // delete a node at end, return true or false

   - int popSList(pSList p); // delete a node at front, return item if successful
     // if statck is empty or error, do nothing and return false
   - ( int pushSList(int item, pSList p); // add a node at front, return true or false
   - ( int enqueueSList(int item, pSList p); // add a node at end, return true or false
   - int dequeueSList(pSList p); // delete a node at front, return item if successful
     // if queue is empty or error, do nothing and return false

   Don't change function signatures and/or return types in SList.h and SList.cpp files provided. My test program may call the same function names and arguments with other (big) data set from my own driver program.
2. SListHWDriver.cpp program is provided for your basic testing. Make a good use of verify() and validateSList() during test.

## Assignment 2 (3 points)

3. Keep the SList tail field updated whenever necessary. Then implement the insert function such that they make a good use of the SList tail information. This piece of information "tail" makes the insert operation possible in O(1) instead of O(n) time complexity experienced in insertEndSList().
   - int insertTailSList(int item, pSList p) // add a node at front, return true or false
     // same functionality as insertEndSList
4. While you make some change in the source code for Assignment 2, don't mess with Assignment 1. From time to time make sure that your Assignment 1 code is not broken. It is good idea that you test them separately.
5. You may figure it out how to use verify.h by yourself since there are some examples shown in the driver file.

## Source files available from Dropbox.

SList.h, verify.h, SListNode.cpp, SListHW.cpp, SListHWDriver.cpp

## Checklist

1. Turn in two files called SListHW.cpp and SListHWDriver.cpp in your dropbox.
   Don't change .h file. Don't turn in .h file. If your source code does not compile with the original ~.h file, your homework will not be graded. [-3.0 point]
2. Remove all the other files from Dropbox except the files requested. [-0.5 point]
3. SListDriver.cpp contains the code that shows me how you tested your code. Test your source code thoroughly as possible. You may use toStringSList(), verify(), validateSList() and/or nthSList() for test. I am going to use my own test data as well in your SListDriver.cpp. Whenever a case does not work, there will be [-0.5] penalty.
4. Each source code must be documented properly that includes author(s), student number, contacts (email or/and phone number), creation date, description, input/output parameters, and example(s). [-1.0 point]
5. Include the following line at the top of your every file with your name signed. [-1.0 point]
   On my honor, I pledge that I have neither received nor provided improper assistance in the completion of this assignment. Signed: _____