A Linked Class

LAB # 9

By

Corey Henry

And

Aaron Shepard

***“On my honor, as a Mississippi State University student, I have neither***

***given nor received unauthorized assistance on this academic work.”***

Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSE-1384-06-201430 Intermediate Computer Programming

Class Section # 6

Jesse Farek

4/14/2015

**Analysis and Conclusions**

The purpose of this lab was to practice with linked lists, we were given a driver that passed us variables and had to create a class to run the program. Over all the program was very easy to make except for the insert position function. We had a lot of difficulty getting that to work. Writing the program became a lot easier if you were drawing out the memory as your wrote the code, or drew out what you wanted it to do and then write the code according to that. if statements were extremely important in this class because there were multiple cases were the list would be not be created yet, so you had to create case variables for the program. If I were to redo this program I would use previous and current statements a lot more, as well as just creating a 3 step condition for the insert variable for if the list was none, if the position was == 0 or the position was greater than 0. Knowing how to traverse through the list became very important, and the class notes and practice we did were a vital part of helping write the lab code. Overall it was a good lab that helped me learn a lot.

Source Code:

#Corey Henry & Aaron Shepard #Date Assigned: 7 apr 15

# #

#Course CSE 1384 Sec 06 #Date Due: 14 apr 15

#File name: Lab

#

#Program description- creates a class that runs the driver for linked lists

#Importing from the ListNode class

from ListNode import \*

class LinkedList():

#Constructs a pointer for the linked list

def \_\_init\_\_(self):

self.head = None

#Inserts an item at the front of the linked list

def insert\_front(self,item):

# if node is empty adds item

if self.head == None:

self.head = ListNode(item)

#adds item when the list isnt empty

else:

temp = ListNode(item)

temp.set\_link(self.head)

self.head = temp

temp = None

#Inserts an item in a particular position of the linked list

def insert\_position(self, item, position):

#Determines if node is empty

if self.head == None:

self.head = ListNode(item)

# inserts the item if the position is at 0

else:

current = self.head

if position == 0:

temp= ListNode(item)

temp.set\_link(self.head)

self.head= temp

temp = None

else:

#inserts the item if the postion is great than 0

for each in range(position):

previous = current

current = current.get\_link()

temp = ListNode(item, current)

previous.set\_link(temp)

#inserts an item at the end of the list

def insert\_end(self, item):

#if the list is empty it add the item to the list

if self.head == None:

self.head = ListNode(item)

# adds item to the list when the list is not empty

else:

current = self.head

temp = ListNode(item)

while current.get\_link() != None:

current = current.get\_link()

current.set\_link(temp)

#deletes the item at the front of the list

def delete\_front(self):

current = self.head

self.head = current.get\_link()

#deletes the item at the end of the list

def delete\_end(self):

current = self.head

while current.get\_link() != None:

previous = current

current = current.get\_link()

previous.set\_link(None)

#deletes an item at a particular position in the list

def delete\_item(self,item):

current = self.head

#if the pointer is equal to item the item is deleted

if current.get\_item() == item:

self.head = self.head.get\_link()

#goes through the list until current is equal to item

else:

while current.get\_item() != item:

previous = current

current = current.get\_link()

previous.set\_link(current.get\_link())

# delete the whole list

def delete\_list(self):

self.head = None

#return string version of the list allowed for None and for elements

def \_\_str\_\_(self):

if self.head == None:

string = 'None'

return string

else:

string = ''

current = self.head

while current != None:

temp = current.get\_item()

string += str(temp)

current = current.get\_link()

if current != None:

string += ', '

else:

string += ''

return string

