**CSE 281: Data structures and Algorithms Lab**

**Singly Linked List**

*Instructions*

* *Write the algorithm and java program codes for the questions from 1 to 13 in the lab record.*
* *Reference to java API is available at : http://192.168.0.48/javadocs/api/index.html*

-------------------------------------------------------------------------------------------------------------------------------

1. Declare a class (Node.java) with class name ***Node***  with two attributes:

(a) An integer variable ***data.***

(b) A Node variable ***next.***

***public class Node{***

***public int data;***

***public Node next;***

***}***

1. Add a default constructor that will assign default values to the Object of type ***Node*** .

***public*** ***Node () {***

***data=0;***

***next=null;***

***}***

1. Add a parameterized constructor that will assign specified values to the Object of type ***Node.***

***public*** ***Node (int x) {***

***data=x;***

***next=null;***

***}***

1. Declare another class (***SinglyLinkedList.java***) with class name ***SinglyLinkedList*** and one attribute ***head*** of type ***Node*** and assign it with ***null***.

***public class SinglyLinkedList {***

***public Node head=null;***

***- --- ------- -----***

***--- -- ---- --- ------***

***}***

1. Implement a method ***insert\_At\_Head()*** inside the class ***SinglyLinkedList.java*** that will insert a Node at the head position your list. It should be public.

***public void insert\_At\_Head(int value)*** ***{***

***Node newnode=new Node(value);// new node creation***

***if( head==null){ //empty Node insertion***

***head=newnode;***

***}***

***else{***

***newnode.next=head;***

***head=newnode;***

***}***

***}***

1. Add print function ***print()*** to ***SinglyLinkedList.java*** that will print the contents of the list It should be public.

***public void print() {***

***if(head==null){***

***System.out.println(“Empty list”);***

***}***

***else{***

***Node temp=head;***

***While(temp!=null){***

***System.out.println(temp.data);***

***temp=temp.next;***

***}***

***}***

***}***

1. Write a test class file ***Test.java .***

***public class Test {***

***public static void main(String args[]){***

***}***

***}***

(a) write the following statements inside the test file that tries to add a new node at the head position.

***SinglyLinkedList ob1 = new SinglyLinkedList ();***

***ob1. insert\_At\_Head (90);***

***ob1.print();***

***ob1. insert\_At\_Head (10);***

***ob1.print();***

Try inserting more elements at the head position.

1. Implement a method ***insert\_At\_Tail()*** inside the class ***SinglyLinkedList.java*** that will insert a new Node at the tail position in your list. It should be public

***public void insert\_At\_Tail(int value)*** ***{***

***Node newnode=new Node(value);// new node creation***

***if( head==null){ //empty node insertion***

***head=newnode;***

***}***

***else{***

***// your logic here***

***}***

***}***

1. Write the following statements inside the test file that tries to add a new node at the tail position.

***SinglyLinkedList ob2 = new SinglyLinkedList ();***

***ob2. insert\_At\_Tail (20);***

***ob2.print();***

***ob2. insert\_At\_Tail (32);***

***ob2.print();***

Try inserting more elements at the tail position.

1. Implement a method ***insert\_At\_position(int value, int k)*** inside the class ***SinglyLinkedList.java*** that will insert a new Node at the position ***k.***

***public void insert\_At\_Tail(int value,int k)*** ***{***

***//your logic here***

***}***

1. Write the corresponding statements inside the test file that tries to add a new node at the ***K th*** position and test it by calling the ***print().***
2. Implement a Delete method ***Delete\_At\_Pos(int k)*** inside the class ***SinglyLinkedList.java*** that will delete a particular element at a given position inside the list. Test this method by writing the corresponding statements inside the test file.

***public void Delete\_At\_Pos(int k)*** ***{***

***//your logic here***

***}***

1. Implement another Delete method ***Delete\_A\_Value(int value)*** by deleting a particular Node based on the value stored in that Node .

***public void Delete\_A\_Value(int value)*** ***{***

***//your logic here***

***}***

1. Implement the following functions (inside the ***SinglyLinkedList.java)*** that formulate the following functions

a. Count the number of nodes in the list.(hint: use a count variable while traversing the list)

b. Returns the sum of elements of the Nodes in the list.

c. Change the data field of a Node with given value.

d. Move the largest element Node to end of the list.

e. To search a Node in the linked list.

f. To concatenate two singly linked lists.

g. To delete even elements in a linked list.

1. Write the java codes to reverse a singly linked list using recursion and without recursion.
2. Implement a ***split*** method that split a singly linked list into two lists so that all elements in odd positions are in one list and those in even positions are in another list.
3. Implement a Stack using singly linked list.
4. Implement a Queue using singly linked list.