**Logo

Description automatically generated San Francisco Bay University**

**CS350 - Data Structures**

**Homework Assignment #2**

**Due day: 2/20/2023**

**Instruction:**

1. **Push the source code to Github**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue homework submission could not be accepted.**

**4. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. Write a function/method to sort a given linked list with *char* type node from *a-zA-Z*. For example, *ls = "Head->D->A->C->A->G->NULL",* the new linked list should be

*"Head->A->A->C->D->G->NULL"* by calling function ***srt\_LL(****ls****)***

2. Assuming that there are two linked lists with positive *number* type value in each node, such as *u= "Head->1->2->3->4->NULL"* and *v="Head->5->6->7->8->NULL",* two numbers *1234* & *5678* from the linked lists can be extracted respectively, and make addition operation for two numbers, the result will be *6912(=1234+5678)* . Find a function/method to implement above operations and get such result *"Head->6->9->1*

*->2->NULL"* if calling function ***LL\_add(****u,v****)***

3. Solve monkey king election question on the handout by circular linked list in a program

*Input*

*Enter total number of monkeys in a group: 5*

*Enter m value: 3*

*Output*

*The king will be 3*

*Input*

*Enter total number of monkeys in a group: 8*

*Enter m value: 5*

*Output*

*The king will be 2*

*Notice that 1st monkey’s label should start with 0 as an index*

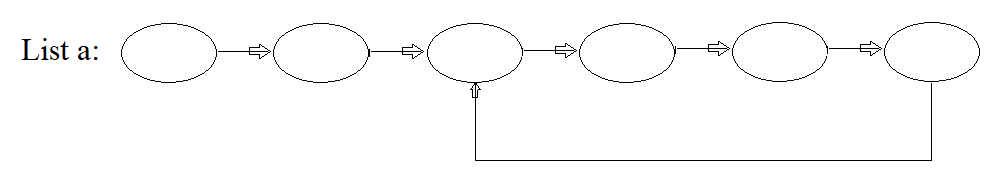
4. Given two singly linked lists with *char* type value node, like *a-zA-Z*

*l= "Head->G->O->O->D->NULL",* and *m ="Head->G->o->O->g->L->E->NULL",* Find the common *char* values and form a new linked list *"Head->G->O->NULL"* through a function/method called ***commElem(****l, m****)****.*

5. Write a function ***diffElem\_LL(****l, m****)*** as similar as above, but to exact all *char* values, which are NOT common ones, and link them together, such as

*"Head->D->o->g->L->E->NULL",*

6. Given a linked list, write a program to check if there exists an internal linked loop or not. For instance, as following, return value of function/method ***is\_loopLL(****a****)*** is true, otherwise false.



*Notice that there is ONLY one link to next in node structure*

7. Implement two-polynomial multiplication operation (convolution) by a program ***conv\_LL(****l, m****)***

e.g.

Hint: take *Polynomial-1.py & Polynomial-2.py* as reference in file

*Linked List Example-2.zip*