**Logo

Description automatically generated San Francisco Bay University**

**CS350 - Data Structures**

**Homework Assignment #4**

**Due day: 3/15/2023**

**Instructions:**

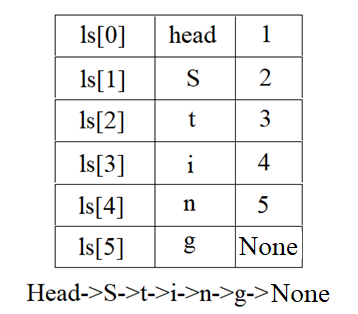
**a. Push the source code to GitHub**

**b. Please follow the code style rule like programs on handout.**

**c. Overdue homework submission can’t be accepted.**

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

1. Linked list can be built in an *array* data type as the following example, in which each element is a class *object* with two components, *data* and *index* pointing to next element in the array.

*import numpy as np*

*class Node:*

*def \_\_init\_\_(self, data, idx):*

*self.data=data*

*self.idx=idx*

*elm0=Node("head",1)*

*elm1=Node("S",2)*

*elm2=Node("t",3)*

*elm3=Node("i",4)*

*elm4=Node("n",5)*

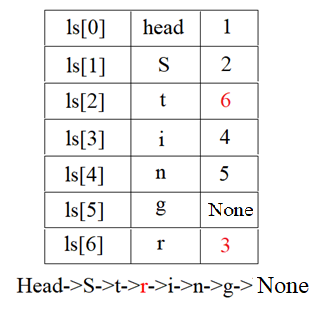
*elm5=Node("g",None)*

*ls=np.array([elm0,elm1,elm2,elm3,elm4,elm5])*

*print(ls[2].data)*

*t*

Write a function/method to insert an element by index to linked list in array structure with a *char* type node. For example, given *ls = Head->S->t->i->n->g->None,* the new one will be *Head->S->t->r->i->n->g->None* by calling function ***insertElem****(ls, 3, 'r')* as shown in the figure below



2. Given a linked list with *int* type node ONLY from *0* to *9*, like

*lst0 = Head->1->2->2->1->None,* and then other three lists can be generated after combinations of any two CONSECUTIVE nodes, such as *lst1 = Head->12->2->1*

*->None, lst2 = Head->1->22->1->None,* and *lst3 = Head->1->2->21->None.* Please generate a function/method to convert those *4* lists to *char* type node linked list based on corresponding English letter *1* to *'A'*, *2* to *'B'*, … … *26* to *'Z'*. If the number is bigger than *26*, replace it with *'#'* character. So outputs will be *Head->A->B->B->A->None* from

*Head->1->2->2->1->None, Head->L->B->A->None* from *Head->12->2->1*

*->None, Head->A->V->A->None* from *Head->1->22->1->None,* and *Head->A->B*

*->U->None* from *Head->1->2->21->None*

3. Given an array with positive integer elements, such as *arr = {1, 3, 2, 1, 2, 1}*, write a program to partition it to two subsets (arrays) with equal sum of all elements, like *ls1={1, 1, 1, 2}* and *ls2 = {2, 3}*, having sum *5* in each array. Notice that the solution may NOT be unique, such as *ls1={3, 1, 1}* and *ls2 = {2, 2, 1}* are other two subsets to meet requirements as well

4. Assuming that all *int* type values are saved in an array, like *a = {3, 9, 10, 1, 30, 40}*, find a function to get maximum value from *a[u] – a[v] + a[w] –a[x]*, where indices must be *u>v>w>x. (40-1+10-3)* will be the result based on given array

5. Given a positive number *N*, *2\*N* elements from *1* to *N* can be created in a list, such as *N=3*, and list will be *1, 1, 2, 2, 3, 3* with two appearances of *1s*, *2s* and *3s*. Write a program to find all sequences between its two appearances with required distance, which is exactly equal to value of the element. For instance, the following two sequences meet the requirements:

e.g. *3 1 2 1 3 2*

*3 … …3 # 3 elements in between*

*1… 1 # 1 element in between*

*2……2 # 2 elements in between*

e.g. *2 3 1 2 1 3*

*2……2 # 2 elements in between*

*3 … …3 # 3 elements in between*

*1… 1 # 1 element in between*

If *N = 4,* two sequences will be

*4 1 3 1 2 4 3 2*

*4 … … 4 # 4 elements in between*

*1… 1 # 1 element in between*

*3 … …3 # 3 elements in between*

*2……2 # 2 elements in between*

*2 3 4 2 1 3 1 4*

*2……2 # 2 elements in between*

*3 … …3 # 3 elements in between*

*4 … … 4 # 4 elements in between*

*1… 1 # 1 element in between*