Assignment 2 (Indicies):

Test completed: 4/11/22

Github link: https://github.com/Vijay-turaka/Advanced_Algorithms-Concordia_Assigments-

(2-1) using 2d matrix:

```
#include <iostream>
using namespace std;

int main()
{
    int n, m, size=0, value, k=0;
    cout<<"enter inputs \n";
    cin>>n;
    cin>>m;
        int sMatrix[n][m];
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < m; j++) {
                    cout<<"enter array values ["<<i<<"] ["<<j<<"] :";
                    cin>>value;
                        if (value != 0) {
                                size++;
                        }
                        sMatrix[i][j]=value;
                }
        }
        int Matrix[3][size];
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < m; j++) {
                        if (sMatrix[i][j] != 0)
                        {
                                Matrix[0][k] = i;
                                Matrix[1][k] = j;
                                Matrix[2][k] = sMatrix[i][j];
                                k++;
                        }
                }
        }

        for (int i=0; i<3; i++)
        {
                for (int j=0; j<size; j++)
                        cout <<Matrix[i][j] <<" ";
                cout <<"\n";
        }
        return 0;
```

```
}
```

(2-2) using Linked lists:

```
#include<iostream>
using namespace std;

class Node {
	public:
	int row;
	int col;
	int data;
	Node *next;
};

void createNode(Node **refFirst, int row, int col, int data) {
	Node *tempFirst = *refFirst;
	Node *newNode;
	if(tempFirst == NULL) {
		tempFirst = new Node();
		tempFirst->row = row;
		tempFirst->col = col;
		tempFirst->data = data;
		tempFirst->next = NULL;
		*refFirst = tempFirst;
	}
	else {
		while(tempFirst->next != NULL) {
			tempFirst = tempFirst->next;
	   }
		newNode = new Node();
		newNode->row = row;
	   newNode->col = col;
		newNode->data = data;
		newNode->next = NULL;
		tempFirst->next = newNode;
	}
}

int main()
{
	int n, m, value, k=0;
	Node *startNode = NULL;
   cout<<"enter size of matrix n and m \n";
   cout<<"enter n value: ";
   cin>>n;
   cout<<"enter m value: ";
   cin>>m;
```

```
        int matrix[n][m];
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < m; j++) {
                    cout<<"enter array values ["<<i<<"] ["<<j<<"] :";
                    cin>>value;
                    matrix[i][j]=value;
                }
        }
        for(int i = 0; i < n; i++) {
                for(int j = 0; j < m; j++) {
                    if(matrix[i][j]!=0) {
                        createNode(&startNode, i, j, matrix[i][j]);
                    }
                }
        }
        cout<< "\nprinting linked list representation:\n";
        while (startNode != NULL)  {
            cout << "* row[" << startNode->row << "]-"<< "col[" << startNode->col << "]-
" << "data[" <<startNode->data <<"]--> ";
                startNode = startNode->next;
        }
        return 0;
}
```

Self evaluation:

1. How long did you spend on this assignment?
    a. 2hr

2. Based on your effort, what letter grade would you say you earned?
    a. On a scale of 1 to 10. I would grade this as 10/10.

3. Based on your solution, what letter grade would you say you earned?
    a. On a scale of 1 to 10. I would grade this as 9/10.

4. Provide a summary of what doesn't work in your solution, along with an explanation of how you attempted to solve the problem and where you feel you struggled?

    a. For matrix approach I used 3*m matrix where each stores row, col, data which makes simple as data increases and easy to traverse
    b. For Linked List approach I create a node object which holds row, col, data as single unit and connect each node to new node.