

**Data Structures BSE**  
**FAST-NU, Lahore, Fall 2022**  
**Assignment 1**

**Due: September 17, 11:59 PM**  
**Marks: 160 pts**  
CLO 3,4,5

Q1. Calculate the time complexity of the following with each step as shown in class. (10 Marks)

<pre>sum = 0; for (i=1;i&lt;=n;i=i*2) {     cout &lt;&lt; i;     cout &lt;&lt; sum;     for (j=1;j&lt;=i;++j)     {         cout &lt;&lt; j;         cout &lt;&lt; "*";         sum++;     }     sum = 0; }</pre>	<pre>for (i=0;i&lt;n;i=i+3) {     cout &lt;&lt; i;     for (j=1;j&lt;n;j=j*3)     {         cout &lt;&lt; j;         sum++;     }     cout &lt;&lt; sum; }</pre>
---	--

Q2. Implement following functions in Linked List (50 Marks total)

- i. **Insert** after the 2<sup>nd</sup> last element of the list (No need to pass index just use some logic) (5Marks)
- ii. **Delete** the 2<sup>nd</sup> node from the list (No need to pass index just use some logic) (5 Marks)
- iii. **Merge**: which merges two linked lists in linear time but do not exclude duplicates (5 Marks)  
For example  
Linked List 1: 10, 20, 30, 40, 50  
Linked List 2: 50, 11, 20, 6  
Merged List: 10, 20, 30, 40, 50, 50, 11, 20, 6
- iv. **Compress**, which removes duplicate nodes from the list. For example (5 Marks)  
Original Linked List: 1, 2, 3, 4, 4, 4, 5, 5, 6, 7, 7, 7, 7, 7, 10  
Linked List after Compress: 1, 2, 3, 4, 5, 6, 7, 10

**Important Note:** User will not enter any data. He will only run your .exe file and it should give following output using the functionality implemented above.

**Insert Functionality:**

**Data: 5**

Linked List 1: 10, 20, 30, 40, 50

After insertion: 10, 20, 30, 40, 5, 50

**Delete Functionality:**

Linked List 1: 10, 20, 30, 40, 5, 50

After deletion: 10, 30, 40, 5, 50

Q3. Write a function to swap nodes in a **Doubly** linked list without swapping data. (You have to make insertion function and print function extra to complete this question) **(15 Marks)**

Input: 10->15->12->13->20->14,                      x = 12, y = 20

Output: 10->15->20->13->12->14

Q4. Convert an Array to a Single Circular Linked List. **(15 Marks)**

i) Write a function to **createLL** to copy array elements into a linklist

ii) Display function for Linklist

Driver Function:

```
int main()
{
    // Array to be converted
    int arr[] = {1,2,3,4,5};
    int s = sizeof(arr) / sizeof(arr[0]);
    createLL(arr, n);

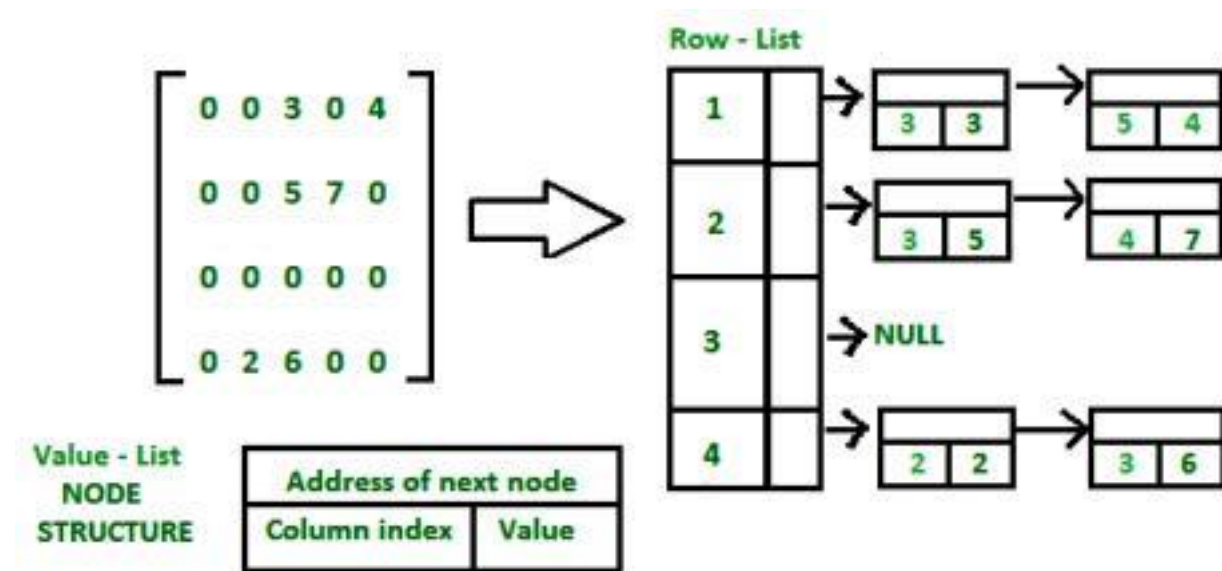
    // Display the list
    print();

    return 0;
}
```

Q5.Many computing applications produce matrices of numbers – integers – in which most of the numbers are 0. Such matrices are called Sparse Matrices.

In an  $N \times N$  matrix, the total number of entries is  $N^2$ . So it requires  $O(N^2)$  space. However, in a Sparse Matrix most of these entries are 0. It is possible that only  $O(N)$  entries are non-zero. So, if we stored only the non-zero entries, we could store the entire matrix in only  $O(N)$  space. Such a matrix can be stored in only  $O(N)$  space by using a scheme which only stores the non-zero elements of each row in a separate linked list (one linked list per row). The following diagram shows the scheme:

(100 Marks)



**Note:** both row numbers and column numbers start from 1.

The structure of each node is shown above. It contains the value of the matrix entry, its column number and a pointer to the next node in the row. Note that the size of each linked list may be different, depending upon how many non-zero entries exist in that row.

Also pay attention to the Row List which contains the head pointers of all the lists. In our implementation, we will use the class vector to maintain these head pointers. The following code shows the basic definition of the class SparseMatrix and the struct Node. Integers M and N contain the number of rows and columns of the matrix respectively. (You will use File handling In this question.)

**Note:** you absolutely must not store the entire  $M \times N$  matrix in a 2D array at any time in your program.

Doing so will result in a 0 in the assignment.

```
class SparseMatrix{
    struct Node{
        int value;
        int colIndex;
        Node* nextInRow;
        Node(int v):value(v){}
    };
    int M, N;//dimensions
    vector<Node*> rowList;
};
```

You are required to add the following public methods to this class:

**1. read(string filename):**

This method should be able to read an  $M \times N$  matrix from a text file. Each line in the file contains all the entries in that row (including the zeros) separated by spaces. There is one row per line. Following is how the matrix shown in the previous picture will be stored in the file:

00304

00570

00000

02600

This file contains 4 lines. The characters in each line are separated by a space. Your function should be able to load this matrix as an object of SparseMatrix, as described above.

## **2. SparseMatrix(*string filename*)**

Constructor. Simply uses read to read the matrix from the specified file.

## **3. SparseMatrix(const SparseMatrix& obj)**

A regular copy constructor creates a deep copy.

## **4. const SparseMatrix & operator = (const SparseMatrix & obj)**

A regular assignment operator creates a deep copy.

## **5. ~SparseMatrix()**

Destructor cleans up all the allocated memory.

**6. SparseMatrix operator + (const SparseMatrix & obj)**

Adds two Sparse Matrices of same dimensions and returns the result in a third Sparse Matrix

**7. SparseMatrix operator \* (const SparseMatrix & obj)**

Multiplies two Sparse Matrices with compatible dimensions for multiplication; say,  $M1 \times N1$  and  $N1 \times L1$ , and returns their  $M1 \times L1$  resultant product Sparse Matrix.

**8. bool isSubMatrix(const SparseMatrix & obj)**

Returns true if the matrix obj is completely contained in the current matrix.

Returns false otherwise.

For example, the matrix  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  is a sub-matrix of  $\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \end{bmatrix}$ .

Again, note that your matrices are stored in the space efficient manner described above and must never be reconstructed into  $M \times N$  arrays at any point in your program.

**9. Inverse of a sparse matrix(minimum of 3 x 3)**

**10. Check whether the sparse matrix is symmetric or skew symmetric.**

**Plagiarism will be checked strictly and if caught, all the assignments of that student will be marked 0.**

\*\*\*\*\* END \*\*\*\*\*