



DATA STRUCTURES

LAB V –Stack using Linked List, Queue using Linked List, Polynomial Expression, Sparse Matrix.

OUTLINE

- Stack using Singly Linked List.
- Queue using Singly Linked List.
- Polynomial Representation using singly linked list
- Polynomial Evaluation using singly linked list
- Exercise
- Prelab Questions

STACK USING SINGLY LINKED LIST

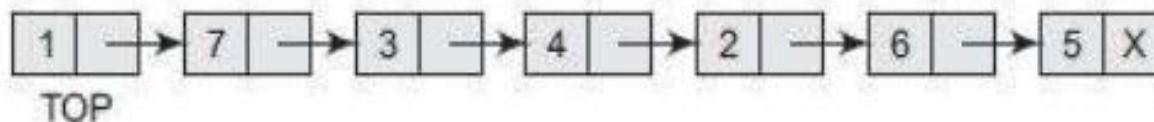
Linked Representation Of Stack

- ❑ The drawback of using array method is fixed size, but in this method it's removed.
- ❑ The storage requirement of linked representation of the stack with n elements is $O(n)$ and the typical time requirement for the operation is $O(1)$.
- ❑ In a linked stack, every node has two parts : one that stores data and another that stores the address of the next node.
- ❑ The START pointer of the linked list is used as TOP.

OPERATIONS ON A LINKED STACK

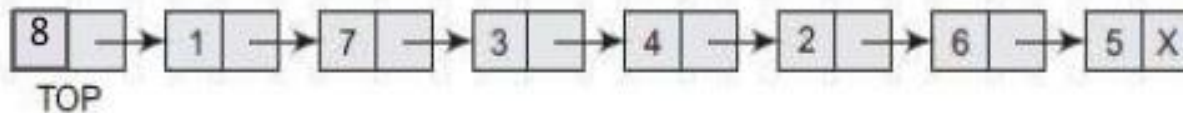
Push Operation:-

- The push operation is used to insert an element into the stack.
- The new element is added at the topmost position of the stack.



Linked Stack

- To insert an element with value 8, we first check if $TOP = NULL$.
- If this is the case, then we allocate memory for a new node, store the value in its DATA part and NULL in its NEXT part.
- The new node will then be called TOP.
- If $TOP \neq NULL$, then we insert the new node at the beginning of the linked stack and name this new node as TOP.

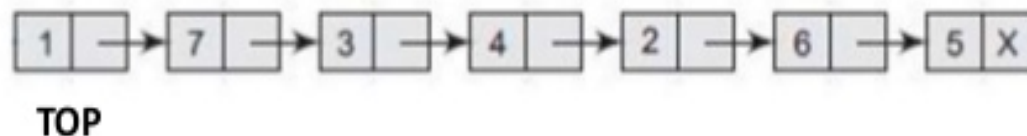


Linked stack after inserting a new node

Pop Operation:-

- The pop operation is used to delete an element into the stack.
- The element is deleted at the topmost position of the stack.
- However, before deleting the value, we must first check if $TOP = NULL$, because if this is the case, then it means that the stack is empty and no more deletions can be done

- If an attempt is made to delete a value from a stack that is already empty, an UNDERFLOW message is printed.
- In case $TOP \neq NULL$, then we will delete the node pointed by TOP, and make TOP point to the second element of the linked stack. Thus, the updated stack becomes like this.



Linked stack after deleting a node

ALGORITHM TO PUSH AN ELEMENT INTO A LINKED STACK

Step 1: Allocate memory for the new node and name it as NEW_NODE

Step 2: SET NEW_NODE -> DATA = VAL

Step 3: IF TOP = NULL

 SET NEW_NODE -> NEXT = NULL

 SET TOP = NEW_NODE

ELSE

 SET NEW_NODE -> NEXT = TOP

 SET TOP = NEW_NODE

[END OF IF]

Step 4: END

- In Step 1, memory is allocated for the new node.
- In Step 2, the DATA part of the new node is initialized with the value to be stored in the node.
- In Step 3, we check if the new node is the first node of the linked list.
- This is done by checking if TOP = NULL. In case the IF statement evaluates to true, then NULL is stored in the NEXT part of the node and the new node is called TOP. However, if the new node is not the first node in the list, then it is added before the first node of the list (that is, the TOP node) and termed as TOP.

ALGORITHM TO POP AN ELEMENT FROM LINKED STACK

```
Step 1: IF TOP=NULL
        PRINT "UNDERFLOW"
        Goto Step 5
    [END OF IF]

Step 2: SET PTR = TOP

Step 3: SET TOP = TOP->NEXT

Step 4: FREE PTR

Step 5:  END
```

- In Step 1, we first check for the UNDERFLOW condition.

TOP!=NULL

- In Step 2, we use a pointer PTR that points to TOP.

- In Step 3, TOP is made to point to the next node in sequence.

- In Step 4, the memory occupied by PTR is given back to the free pool.

Peek Operation:-

- The Peek operation is used to see the value of the topmost element of the stack without deleting it from the stack.
- Peek operation first checks if the stack is empty, i.e., if $TOP = NULL$, then an appropriate message is printed, else the value is returned.

Algorithm

To Peek an element from a linked stack

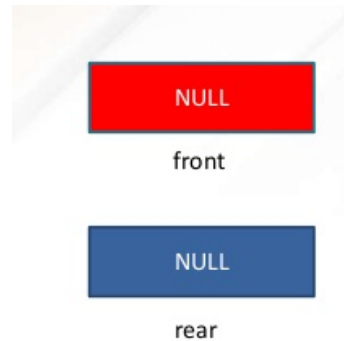
```
Step 1: IF TOP = NULL
        PRINT "UNDERFLOW"
    ELSE
        RETURN TOP->DATA
    [END OF IF]
Step 2: END
```

- In step 1, if TOP = NULL means that the stack is empty.
- Else it will return the data on TOP, which will be our output or Peeped out Value.

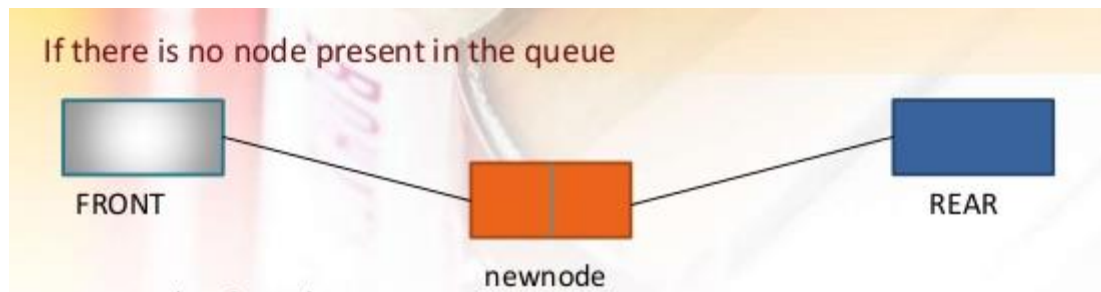
QUEUE USING SINGLY LINKED LIST

ENQUEUE OPERATION

- At the beginning set front and rear to null.

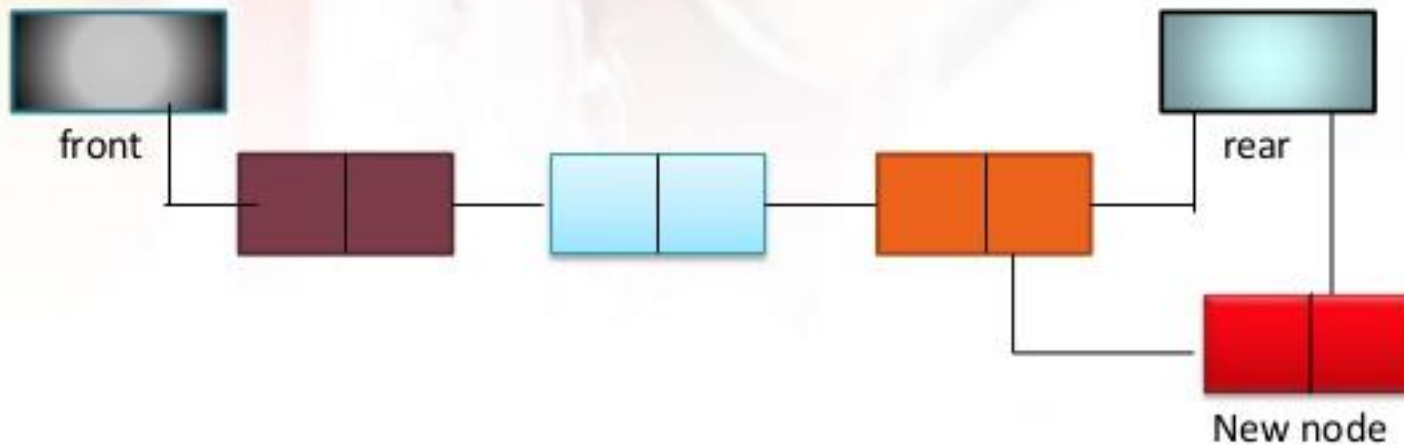


- If there is no node then set front and rear is equal to new node:



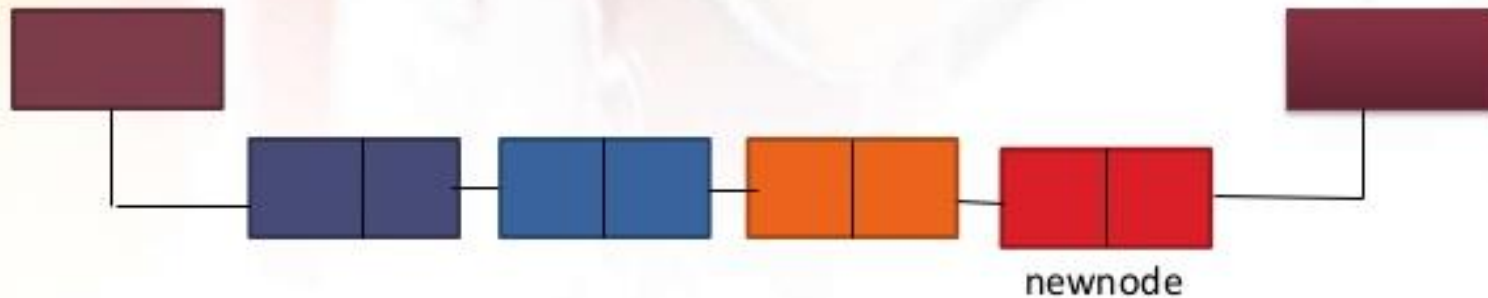
ENQUEUE

- If there is one node and multiple node in the queue



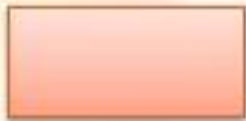
AFTER INSERTION OF NODE IN THE QUEUE

- If there is one node and multiple node in the queue



DEQUEUE

- If there is no node in the queue.
- Message should be prompted to the user.

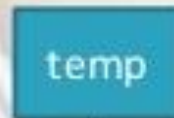


front



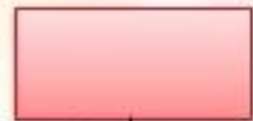
rear

- If there is one node available in the queue.



temp

temp=ptrf



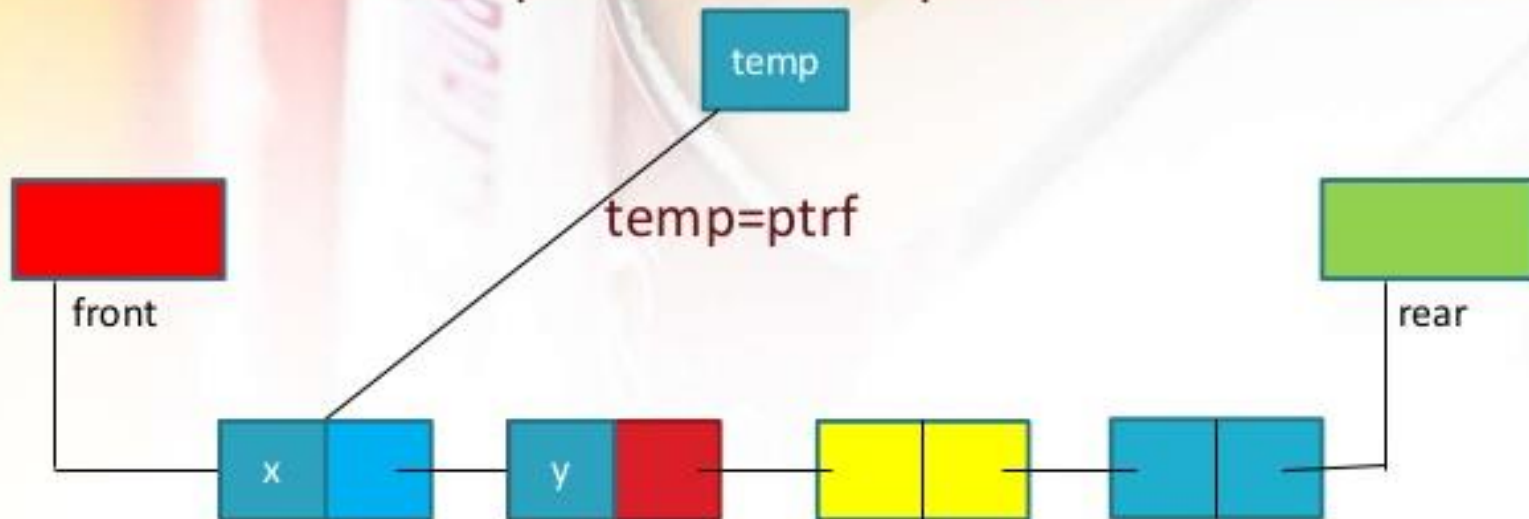
front



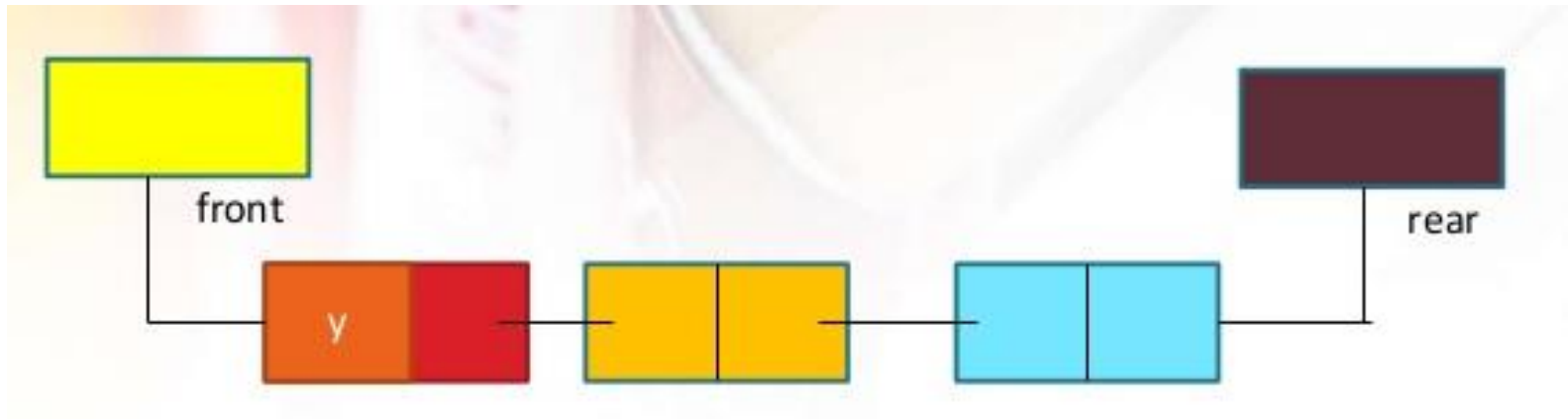
rear

DEQUEUE

- If there are multiple node in the queue.



AFTER DELETION FROM QUEUE



POLYNOMIAL REPRESENTATION

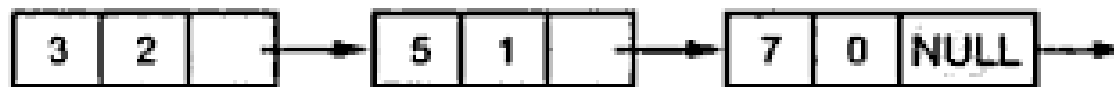
- A polynomial $p(x)$ is an expression in variable x which is of the form $(ax^n + bx^{n-1} + \dots + jx + k)$, where a, b, c, \dots, k fall in the category of real numbers and ' n ' is non-negative integer, which is called the degree of polynomial.
- An essential characteristic of the polynomial is that each term in the polynomial expression consists of two parts:
 - one is the coefficient
 - other is the exponent
- An example of polynomial is $p(x)=4x^3+6x^2$ where 4 and 6 are coefficients and 3 and 2 are exponents.
- A polynomial may be represented using array or linked list.

REPRESENTATION OF POLYNOMIAL USING SINGLY LINKED LIST

- A polynomial can be thought of as an ordered list of nonzero terms. Each nonzero term is a two-tuple which holds two pieces of information:
 - The exponent part
 - The coefficient part
- Node representation of a polynomial nonzero term:

Coefficient	Exponent	Address to next node
-------------	----------	----------------------

- In each node the exponent field will store the corresponding exponent and the coefficient field will store the corresponding coefficient. Link field points to the next item in the polynomial.
- For example $3x^2 + 5x + 7$ will represent as follows.



OPERATIONS

- Expression Evaluation
- Addition
- Multiplication

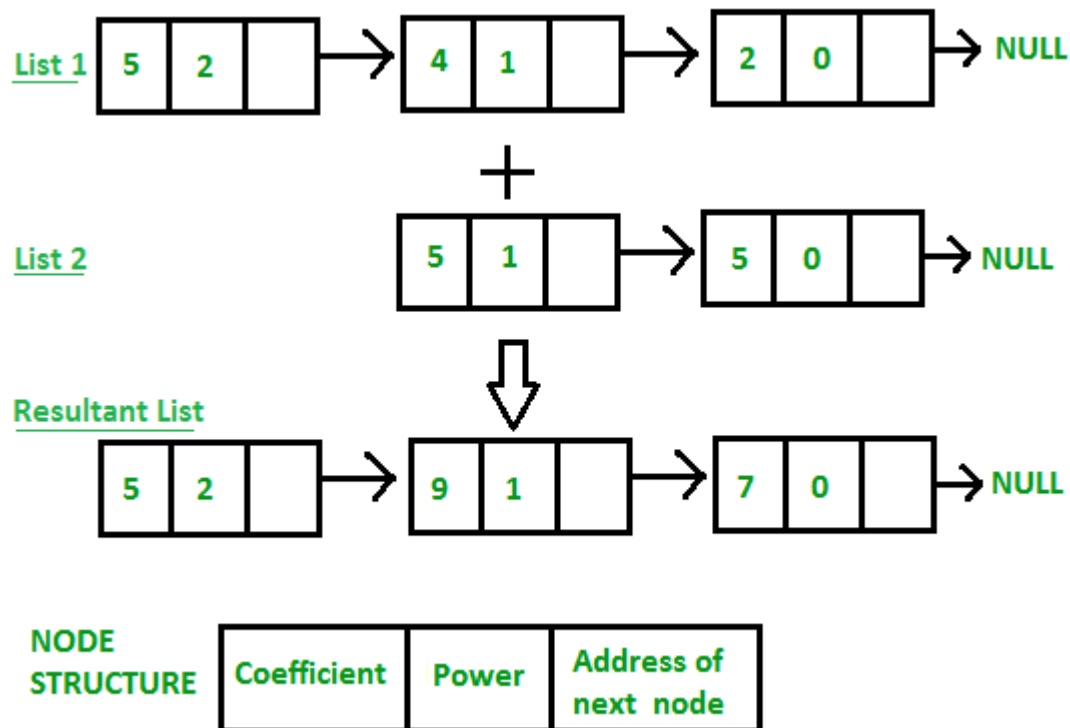
POLYNOMIAL EXPRESSION EVALUATION

- Evaluate the given polynomial expression by read the variable value.
- For example:
- If $p(x) = 3x^2 + 5x + 7$ and x value read is 3, then the result of the polynomial expression is

$$\begin{aligned}p(3) &= 3*3^2 + 5*3 + 7 \\&= 3*9 + 5*3 + 7 \\&= 18 + 15 + 7 \\&= 40\end{aligned}$$

ADDING TWO POLYNOMIALS USING LINKED LIST

- Given two polynomial numbers represented by a linked list. Addition of two polynomials can be represented as follows:



PROCEDURE FOR ADDITION

Step 1: Loop around all values of linked list and follow step 2& 3.

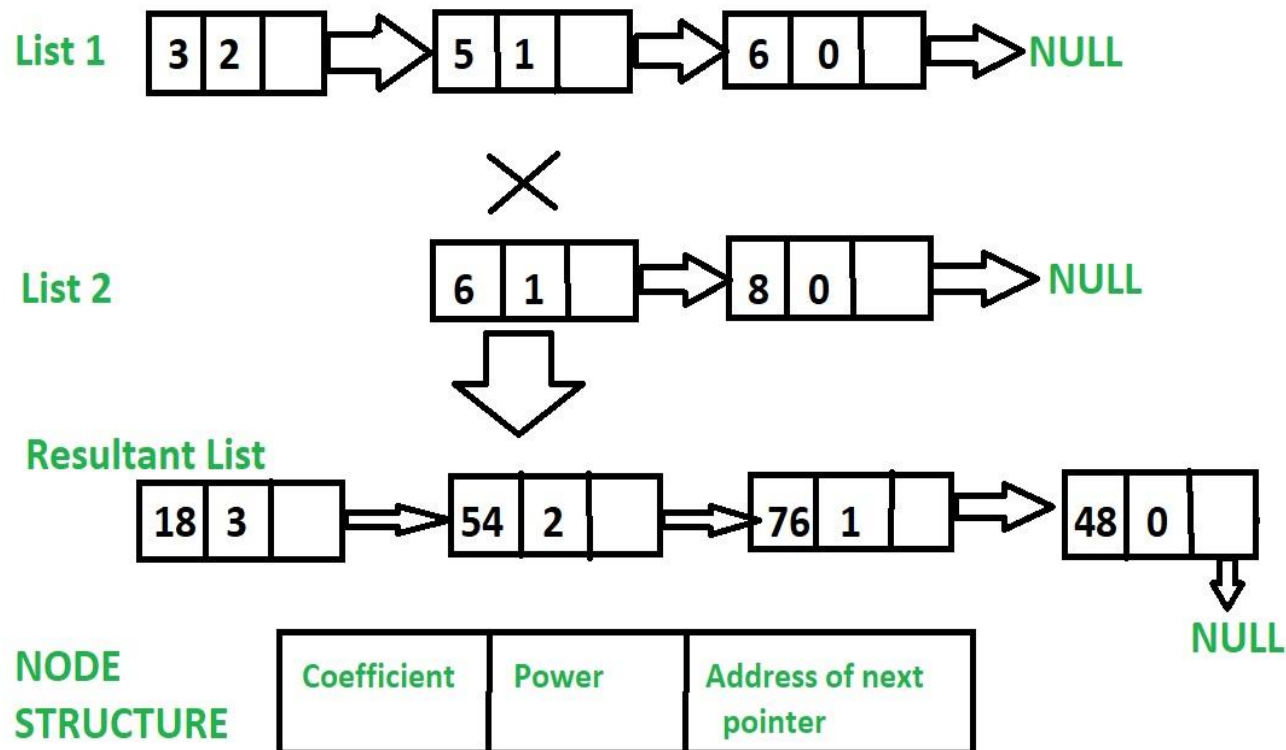
Step 2: If the value of a node's exponent is greater copy this node to result node and move towards the next node.

Step 3: if the values of both node's exponent is same add the coefficients and then copy the added value with node to the result and move towards the next node in both the lists.

Step 4: Print the resultant node.

POLYNOMIAL EXPRESSION MULTIPLICATION

- Given two polynomial numbers represented by a linked list. Addition of two polynomials can be represented as follows:



PROCEDURE FOR MULTIPLICATION

- In polynomial multiplication, we will multiply the 2nd polynomial with each term of 1st polynomial.
- Store the multiplied value in a new linked list.
- Then we will add the coefficients of elements having the same power in resultant polynomial.

SPARSE MATRIX

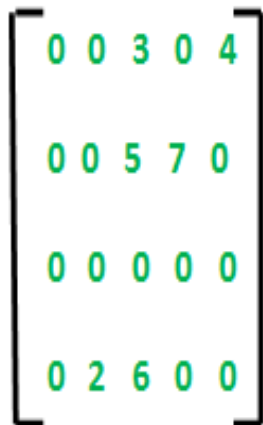
- A matrix is a two-dimensional data object made of m rows and n columns, therefore having total m x n values. If most of the elements of the matrix have 0 value, then it is called a sparse matrix.
- Why to use Sparse Matrix instead of simple matrix ?
 - Storage: There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
 - Computing time: Computing time can be saved by logically designing a data structure traversing only non-zero elements.
- Example:

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$

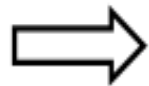
- So, instead of storing zeroes with non-zero elements, we only store non-zero elements. This means storing non-zero elements with triples- (Row, Column, value).

SPARSE MATRIX REPRESENTATION USING ARRAYS

- Sparse Matrix Representations can be done in many ways following are two common representations:
- 2D array is used to represent a sparse matrix in which there are three rows named as
 - Row: Index of row, where non-zero element is located
 - Column: Index of column, where non-zero element is located
 - Value: Value of the nonzero element located at index – (row, column)



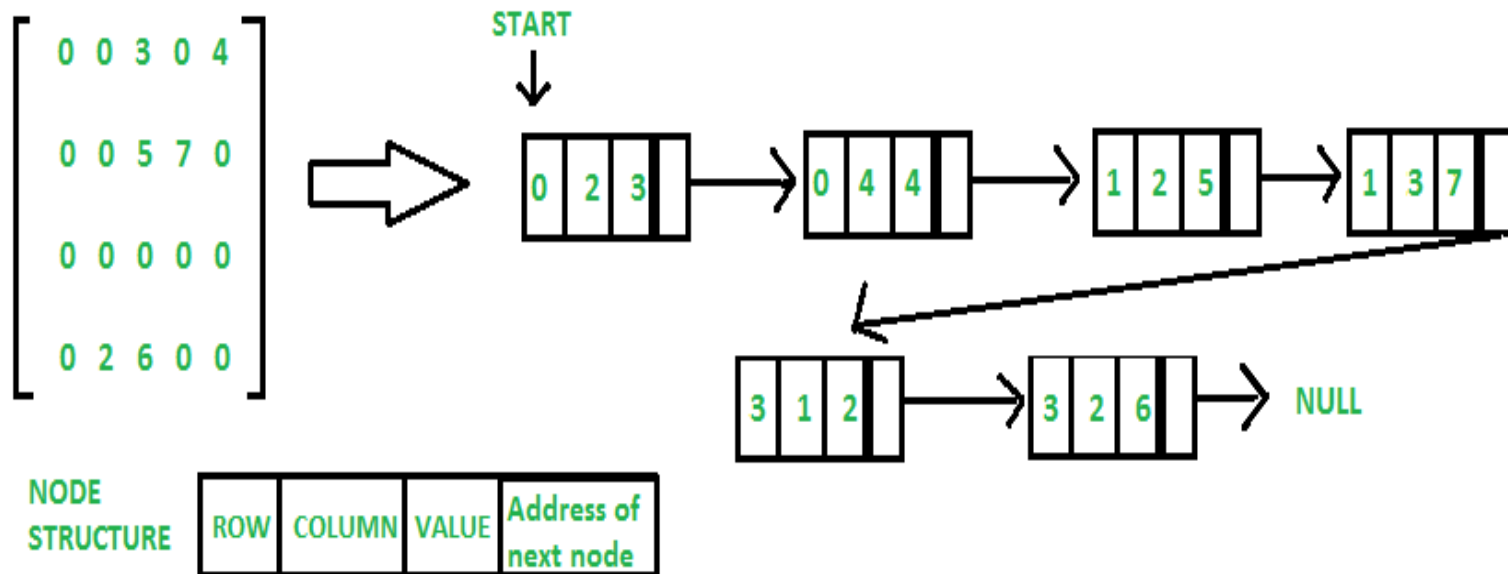
0	0	3	0	4
0	0	5	7	0
0	0	0	0	0
0	2	6	0	0



Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

SPARSE MATRIX REPRESENTATION USING SINGLY LINKED LIST

- In linked list, each node has four fields. These four fields are defined as:
 - Row: Index of row, where non-zero element is located
 - Column: Index of column, where non-zero element is located
 - Value: Value of the non zero element located at index – (row,column)
 - Next node: Address of the next node



OPERATIONS ON SPARSE MATRIX

- Given two sparse matrices, we can perform operations such as add, multiply or transpose of the matrices in their sparse form itself.
- Procedure for Addition:
 - The sparse matrix used anywhere in the program is sorted according to its row values. Two elements with the same row values are further sorted according to their column values.
 - Now to **Add** the matrices, we simply traverse through both matrices element by element and insert the smaller element (one with smaller row and col value) into the resultant matrix. If we come across an element with the same row and column value, we simply add their values and insert the added data into the resultant matrix.

EXERCISE

- Write a program to implement Stack using singly linked list.
- Write a program to implement Queue using singly linked list.
- Write a menu driven C program to implement the following polynomial operations using singly linked list.
 - a) Expression evaluation
 - b) Polynomial Addition
 - c) Display
- Write a C program to implement sparse matrix addition using singly linked list.

PRELAB QUESTIONS

- How will you find the middle of the linked list.
- Write a logic to insert an element in the sorted linked list.
- How will you display a linked list from end.
- Compare polynomial representation between arrays and linked list.
- Analyze the time complexity for each of the following operations:
 - A) poly creation
 - B) poly addition
 - C) poly evaluation