**Data Structures Lab 05(b)**

**Course:** Data Structures (CL2001) **Semester:** Fall 2024

**Instructor: Sameer Faisal T.A:** N/A

**Note:**

* + - * Lab manual cover following below elementary sorting algorithms
* Maintain discipline during the lab.
* Just raise hand if you have any problem.
* Get your lab checked at the end of the session.

**Stack with Array**

class Stack {

    int top;

public:

    int a[MAX]; // Maximum size of Stack

    Stack() { top = -1; }

    bool push(int x);

    int pop();

    int peek();

    bool isEmpty();

};

bool Stack::push(int x)

{

    if (top >= (MAX - 1)) {

        cout << "Stack Overflow";

        return false;

    }

    else {

        a[++top] = x;

        cout << x << " pushed into stack\n";

        return true;

    }

}

int Stack::pop()

{

    if (top < 0) {

        cout << "Stack Underflow";

        return 0;

    }

    else {

        int x = a[top--];

        return x;

    }

}

int Stack::peek()

{

    if (top < 0) {

        cout << "Stack is Empty";

        return 0;

    }

    else {

        int x = a[top];

        return x;

    }

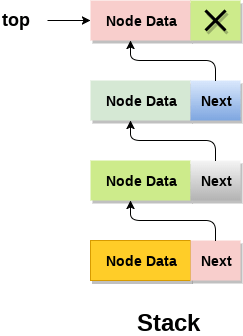
}

bool Stack::isEmpty()

{

    return (top < 0);

}

**Stack with Linked list**

struct Node

{

int data;

struct Node\* link;

};

struct Node\* top;

// Utility function to add an element

// data in the stack insert at the beginning

void push(int data)

{

// Create new node temp and allocate memory

struct Node\* temp;

temp = new Node();

// Check if stack (heap) is full.

// Then inserting an element would

// lead to stack overflow

if (!temp)

{

cout << "\nHeap Overflow";

exit(1);

}

// Initialize data into temp data field

temp->data = data;

// Put top pointer reference into temp link

temp->link = top;

// Make temp as top of Stack

top = temp;

}