CL-2001 Data Structures – Lab

Objectives:

- **Stack implementations** i.e. both Array & LinkedList based with Applications of stacks.

Instructions:

- Write code with proper indentation and use brief comments to explain your code.
- Upload both the files (.**cpp** files + word file) code with screenshots of every task output.
- **Plagiarism** from any source or your friends will be penalized with **ZERO** marks to both parties.

```
class IntStack
{
    private:
        int *stackArray;
        int stackSize;
        int top;
    public:
        IntStack(int)  {}
        ~IntStack()    {}
        bool push(int) {}
        bool pop(int &) {}
        bool isFull()  {}
        bool isEmpty() {}
        int top() {}    // return top of stack
};
```

```
class IntStack
{
private:
    node*top;
public:
    IntStack( ) {}
    ~IntStack( ) {}
    bool push(int) {}
    bool pop(int &) {}
    bool isEmpty() {}
    int top() {}
};
```

Problem#01: Stack ADT (easy)

You are required to implement **Stack ADT** both with **Array & LinkedList** based data structures. Basic operations are mentioned in the above image.

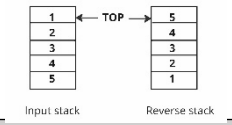
Problem#02: Stack ADT – Array based (easy)

Use the Array based Stack ADT implemented above, add the following function inside class and implement it.

Int **getMin()**: returns minimum element of the stack in **O(1)** time.

Problem#03: Reverse a **Stack** (easy)

Write a C++ program to **reverse** the content of an array-based stack. You're only allowed to use stacks for any purpose i.e. temporary storage etc.



Problem#04: Brackets Matching (medium)

Write a program using above stack ADTs to check if a given string containing different types of brackets (like (), {}, and []) is balanced. Ignore non-bracket characters if any.

Sample Input: "{ [( ) ( )] }" Sample Output: The string **is balanced**.

Sample Input: "{ [( ]) }" Sample Output: The string is **not balanced**.

Problem#05: Evaluate Postfix Expression (easy)

Write a program to evaluate a postfix expression using a stack.

Sample Input: "5 6 2 + * 12 4 / -" Sample Output: 27

Sample Input: "2 3 1 * + 9 -" Sample Output: -4

Problem#06: Palindrome (Medium)

Write a function to check if a given string is a palindrome using a stack. Ignore non-alphanumeric characters and treat upper and lowercase letters as equal.

Sample Input: "A man, a plan, a canal, Panama!"

Sample Output: The string is **a palindrome**.

Sample Input: "Hello, World!"

Sample Output: The string is **not a** palindrome.

Problem#07: Sort Stack (Medium)

Write a recursive algorithm to sort a stack (in ascending order). You cannot use any additional data structures, but recursion is allowed.