

LAB 08

Miss Atika Aslam

Muhammad Abdullah

53457

### Lab Task 1:

Write a program to implement a stack using a linked list. Include the following operations:

* Push (Insert an element at the top of the stack)
* Pop (Remove the top element of the stack)
* Peek (Return the top element without removing it)
* Check if the stack is empty

**Code:**

#*include* <iostream>

using namespace std;

class Stack {

private:

    struct Node {

        int data;

        Node\* next;

*Node*(int value) : *data*(value), *next*(nullptr) {}

    };

    Node\* top;

public:

*Stack*() : *top*(nullptr) {}

    void *push*(int value) {

        Node\* newNode = new *Node*(value);

        newNode->next = top;

        top = newNode;

    }

    void *pop*() {

*if* (*isEmpty*()) {

            cout *<<* "Stack is empty. Cannot pop.\n";

*return*;

        }

        Node\* temp = top;

        top = top->next;

        delete temp;

    }

    int *peek*() {

*if* (*isEmpty*()) {

            cout *<<* "Stack is empty.\n";

*return* -1;

        }

*return* top->data;

    }

    bool *isEmpty*() {

*return* top == nullptr;

    }

    void *display*() {

        Node\* temp = top;

*while* (temp) {

            cout *<<* temp->data *<<* " ";

            temp = temp->next;

        }

        cout *<<* *endl*;

    }

};

int *main*() {

    Stack s;

    s.*push*(10);

    s.*push*(20);

    s.*push*(30);

    cout *<<* "Stack elements: ";

    s.*display*();

    cout *<<* "Top element: " *<<* s.*peek*() *<<* *endl*;

    s.*pop*();

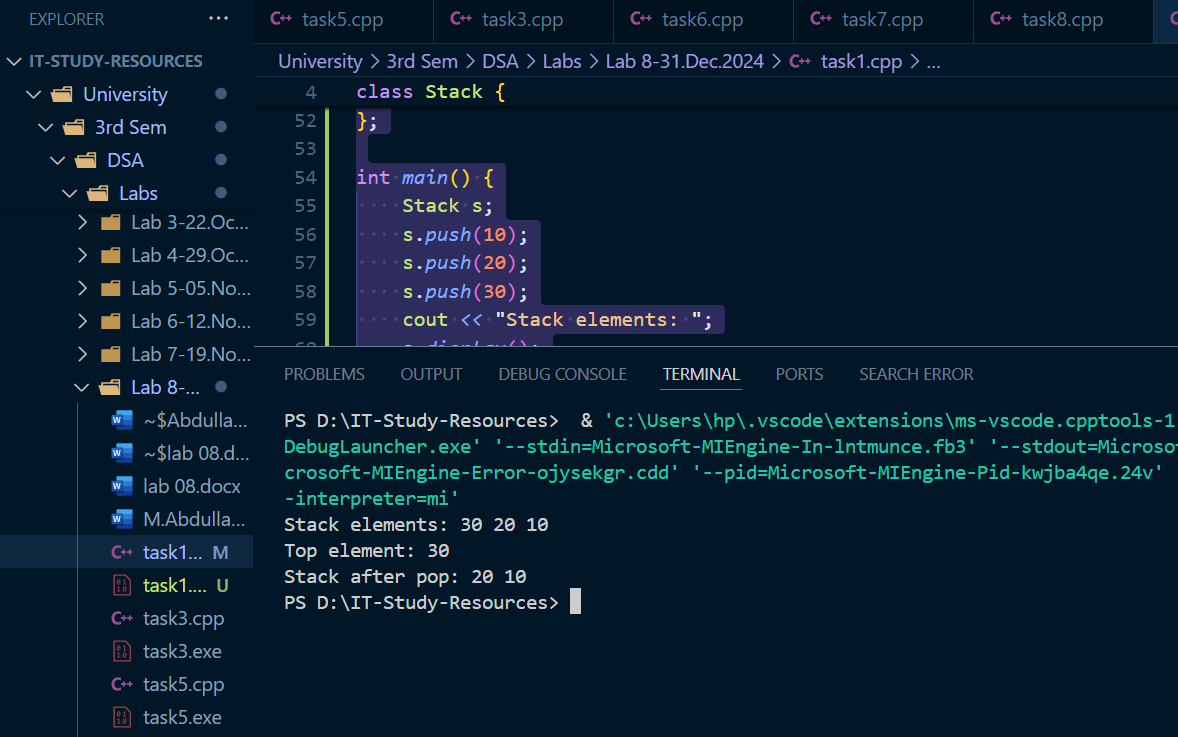
    cout *<<* "Stack after pop: ";

    s.*display*();

*return* 0;

}

## Output



**Lab Task 2:**

Use a stack (implemented using a linked list) to reverse a given string.

* Input: "hello"
* Output: "olleh"

**Code:**

#*include* <iostream>

#*include* <string>

using namespace std;

class Stack {

private:

    struct Node {

        char data;

        Node\* next;

*Node*(char value) : *data*(value), *next*(nullptr) {}

    };

    Node\* top;

public:

*Stack*() : *top*(nullptr) {}

    void *push*(char value) {

        Node\* newNode = new *Node*(value);

        newNode->next = top;

        top = newNode;

    }

    void *pop*() {

*if* (*isEmpty*()) *return*;

        Node\* temp = top;

        top = top->next;

        delete temp;

    }

    char *peek*() {

*return* *isEmpty*() ? '\0' : top->data;

    }

    bool *isEmpty*() {

*return* top == nullptr;

    }

};

string *reverseString*(*const* string*&* str) {

    Stack s;

*for* (char ch : str) s.*push*(ch);

    string reversed = "";

*while* (!s.*isEmpty*()) {

        reversed *+=* s.*peek*();

        s.*pop*();

    }

*return* reversed;

}

int *main*() {

    string input = "Abdullah";

    cout *<<* "Original: " *<<* input *<<* *endl*;

    cout *<<* "Reversed: " *<<* *reverseString*(input) *<<* *endl*;

*return* 0;

}

**OUTPUT**



**Lab Task 3:**

Write a program to check if a string containing parentheses ({}, [], ()) is balanced. Use a stack implemented with a linked list.

* Input: "({[()]})"
* Output: Balanced
* Input: "({[([)])}"
* Output: Not Balanced

**Code:**

**OUTPUT**

**Lab Task 4:**

Sort a stack using recursion and only one additional stack for temporary storage. Implement the stack using a linked list.

* Input Stack: [3, 1, 4, 2]
* Output Stack: [1, 2, 3, 4]

**Code:**

**OUTPUT**

**Lab Task 5:**

Use a stack (implemented using a linked list) to evaluate a postfix expression.

* Input: "231\*+9-"
* Output: -4

**Code:**

**OUTPUT**

**Lab Task 6:**

Modify the stack implementation (using a linked list) to include a getMin() function, which returns the minimum element in the stack in O(1) time.

**Code:**

**OUTPUT**

**Lab Task 7:**

Reverse a given linked list using a stack (implemented with another linked list).

* Input: 1 → 2 → 3 → 4
* Output: 4 → 3 → 2 → 1

**Code:**

**OUTPUT**

**Lab Task 8:**

Given an array, find the next greater element for each element using a stack implemented with a linked list.

* Input: [4, 5, 2, 10]
* Output: [5, 10, 10, -1]

**Code:**

**OUTPUT**