# Mahad Ghauri

233523

DSA Lab Task

#include <iostream>

using namespace std;

class Stack

{

private:

    int \*arr;    // Array to store stack elements

    int top;     // Index of the top element

    int maxSize; // Maximum size of the stack

public:

    // Constructor to initialize the stack

    Stack(int size)

    {

        arr = new int[size];

        maxSize = size;

        top = -1;

    }

   // Push an element onto the stack

    void push(char value)

    {

        if (top == maxSize - 1)

        {

            cout << "Stack overflow. Unable to push " << value << "\n";

            return;

        }

        arr[++top] = value;

    }

    // Pop an element from the stack

    char pop()

    {

        if (isEmpty())

        {

            cout << "Stack underflow. No element to pop.\n";

            return -1;

        }

        return arr[top--];

    }

    // Peek at the top element of the stack

    char peek() const

    {

        if (isEmpty())

        {

            return -1; // Returnkking -1 if stack is empty

        }

        return arr[top];

    }

    // Check if the stack is empty

    bool isEmpty() const

    {

        return top == -1;

    }

    // Destructor to free allocated memory

    ~Stack()

    {

        delete[] arr;

    }

    void Push\_And\_Pop(string other)

    {

        for (int i = 0; i < other.length(); i++)

        {

            char token = other[i];

            if (!isdigit(token))

            {

                if (!ispunct(token))

                {

                    cout << "\n Pushed: " << token << " in the stack " << endl;

                    arr[++top] = token;

                }

                else if (token == '\*')

                {

                    cout << "\n Poped '\*' out of the stack " << endl;

                    arr[top--];

                }

            }

        }

    }

};

int main()

{

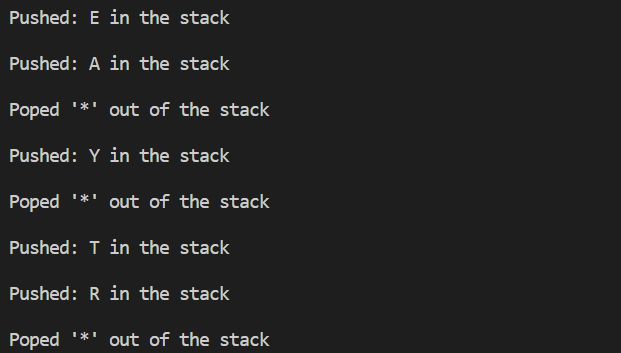
    string name = "EA\*Y\*TR\*";

    Stack s(name.length());

    s.Push\_And\_Pop(name);

    return 0;

}



#include <iostream>

#include <string>

using namespace std;

class Node

{

private:

    int data;

    Node \*next;

public:

    Node(int x)

    {

        data = x;

        next = NULL;

    }

    void setNext(Node \*other)

    {

        next = other;

    }

    void setData(int x)

    {

        data = x;

    }

    Node \*getNext()

    {

        return next;

    }

    int getData()

    {

        return data;

    }

};

class List

{

public:

    Node \*head = NULL;

    Node \*tail = NULL;

    List()

    {

        head = NULL;

        tail = NULL;

    }

    void AddNode(int value)

    {

        if (head == NULL)

        {

            head = new Node(value);

            tail = head;

        }

        else

        {

            tail->setNext(new Node(value));

            tail = tail->getNext();

        }

    }

    void Display()

    {

        Node \*temp = head;

        cout << " List: ";

        while (temp != NULL)

        {

            cout << temp->getData() << " ";

            temp = temp->getNext();

        }

        cout << endl;

    }

    int RemoveNode()

    {

        if (head == NULL)

        {

            cout << "List is empty. No element to dequeue.\n";

            return -1;

        }

        int value = head->getData();

        Node \*temp = head;

        head = head->getNext();

        delete temp;

        return value;

    }

    int Peek()

    {

        if (head == NULL)

        {

            cout << "List is empty. No element to peek.\n";

            return -1;

        }

        return head->getData();

    }

    bool isEmpty()

    {

        return head == NULL;

    }

};

class Queue

{

private:

    List l1;

public:

    Queue()

    {

        // Default Constructor

    }

    void Enqueue(int value)

    {

        l1.AddNode(value);

        cout << " Enqueued: " << value << " to queue.\n";

    }

    int Dequeue()

    {

        return l1.RemoveNode();

    }

    int Peek()

    {

        return l1.Peek();

    }

    bool isEmpty()

    {

        return l1.isEmpty();

    }

    bool isFull()

    {

        return false;

    }

};

int main()

{

    Queue q;

    q.Enqueue(10);

    q.Enqueue(20);

    q.Enqueue(30);

    cout << "Front element: " << q.Peek() << endl;

    cout << " Dequeued " << q.Dequeue() << " from queue." << endl;

    cout << "Front element: " << q.Peek() << endl;

    cout << " Dequeued " << q.Dequeue() << " from queue." << endl;

    cout << " Dequeued " << q.Dequeue() << " from queue." << endl;

    cout << "Is queue empty ? ";

    if (q.isEmpty())

    {

        cout << " Yes " << endl;

    }

    else

    {

        cout << " No " << endl;

    }

    return 0;

}

