

**DSA (Data Structure and Algorithms) Lab**

**LAB REPORT # 8**

**Semester**: 3rdSemester

**Section**: C

**Submitted To:**

**Abdullah Shahrose**

**Submitted By:**

**Name**: Abdul Ahad

**Roll No**: 22-CS-071

**Task 1:**

#include <iostream>

class Node

{

public:

    int data;

    Node \*next;

    Node(int value)

    {

        data = value;

        next = nullptr;

    }

};

class Queue

{

private:

    Node \*front;

    Node \*rear;

public:

    Queue()

    {

        front = nullptr;

        rear = nullptr;

    }

    bool isEmpty()

    {

        return front == nullptr;

    }

    void enqueue(int value)

    {

        Node \*newNode = new Node(value);

        if (isEmpty())

        {

            front = newNode;

            rear = newNode;

        }

        else

        {

            rear->next = newNode;

            rear = newNode;

        }

        std::cout << "Element " << value << " enqueued successfully." << std::endl;

    }

};

int main()

{

    Queue queue;

    queue.enqueue(10);

    queue.enqueue(20);

    queue.enqueue(30);

    return 0;

}

**Output:**



**Task 2:**

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

class Queue {

private:

    Node\* front;

    Node\* rear;

public:

    Queue() {

        front = nullptr;

        rear = nullptr;

    }

    void enqueue(int value) {

        Node\* newNode = new Node;

        newNode->data = value;

        newNode->next = nullptr;

        if (rear == nullptr) {

            front = newNode;

            rear = newNode;

        } else {

            rear->next = newNode;

            rear = newNode;

        }

        std::cout << "Element enqueued: " << value << std::endl;

    }

    void dequeue() {

        if (front == nullptr) {

            std::cout << "Queue is empty. Cannot dequeue." << std::endl;

            return;

        }

        Node\* temp = front;

        int value = temp->data;

        if (front == rear) {

            front = nullptr;

            rear = nullptr;

        } else {

            front = front->next;

        }

        delete temp;

        std::cout << "Element dequeued: " << value << std::endl;

    }

};

int main() {

    Queue queue;

    queue.enqueue(10);

    queue.enqueue(20);

    queue.enqueue(30);

    queue.dequeue();

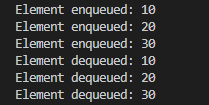
    queue.dequeue();

    queue.dequeue();

    return 0;

}

**Output:**



**Task 3:**

#include <iostream>

class Node

{

public:

    int data;

    Node \*next;

};

class Queue

{

private:

    Node \*front;

    Node \*rear;

    int size;

public:

    Queue()

    {

        front = nullptr;

        rear = nullptr;

        size = 0;

    }

    ~Queue()

    {

        while (!isEmpty())

        {

            dequeue();

        }

    }

    bool isEmpty()

    {

        return (front == nullptr);

    }

    void enqueue(int element)

    {

        Node \*newNode = new Node();

        newNode->data = element;

        newNode->next = nullptr;

        if (isEmpty())

        {

            front = newNode;

            rear = newNode;

        }

        else

        {

            rear->next = newNode;

            rear = newNode;

        }

        size++;

        std::cout << "Element " << element << " enqueued successfully." << std::endl;

    }

    void dequeue()

    {

        if (isEmpty())

        {

            std::cout << "Queue is empty. Cannot dequeue." << std::endl;

            return;

        }

        Node \*temp = front;

        int element = temp->data;

        front = front->next;

        delete temp;

        size--;

        std::cout << "Element " << element << " dequeued successfully." << std::endl;

    }

    int getSize()

    {

        return size;

    }

};

int main()

{

    Queue queue;

    int choice, element;

    do

    {

        std::cout << "1. Enqueue" << std::endl;

        std::cout << "2. Dequeue" << std::endl;

        std::cout << "3. Get Size" << std::endl;

        std::cout << "0. Exit" << std::endl;

        std::cout << "Enter your choice: ";

        std::cin >> choice;

        switch (choice)

        {

        case 1:

            std::cout << "Enter element to enqueue: ";

            std::cin >> element;

            queue.enqueue(element);

            break;

        case 2:

            queue.dequeue();

            break;

        case 3:

            std::cout << "Size of the queue: " << queue.getSize() << std::endl;

            break;

        case 0:

            std::cout << "Exiting..." << std::endl;

            break;

        default:

            std::cout << "Invalid choice. Please try again." << std::endl;

            break;

        }

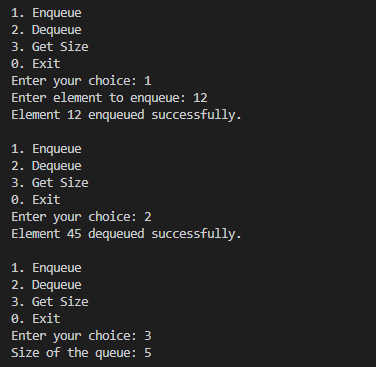
        std::cout << std::endl;

    } while (choice != 0);

    return 0;

}

**Output:**

****