1

**DSA Lab: 09**

****

**Department of Computer Science**

**Iqra University Islamabad**

**DSA**

**Maqsood Ahmed**

**ID: 38186**

**Queue Data Structure using Arrays:**

**Source Code:**

#include <iostream>

using namespace std;

class Queue {

private:

int\* arr;

int front;

int rear;

int size;

int itemCount;

public:

// Constructor

Queue() {

size = 100;

arr = new int[size];

front = rear = -1;

itemCount = 0;

}

// Destructor

~Queue() {

delete[] arr;

}

// Enqueue operation

void enqueue(int value) {

if(isFull()) {

cout << "The queue is Full.\n";

return;

}

if (isEmpty()) front = 0; // Ensure front is updated for the first element

rear = (rear + 1) % size;

arr[rear] = value;

itemCount++;

}

// Dequeue operation

int dequeue() {

if(isEmpty()) {

cout << "The queue is Empty!\n";

return -1;

}

int result = arr[front];

if (front == rear) { // Reset queue if it becomes empty

makeNull();

} else {

front = (front + 1) % size;

itemCount--;

}

return result;

}

// Peek operation

int peek() {

if(isEmpty()) {

throw runtime\_error("Queue is empty!\n");

}

return arr[front];

}

// Check if the queue is full

bool isFull() {

return ((rear + 1) % size) == front;

}

// Check if the queue is empty

bool isEmpty() {

return front == -1;

}

// Reset the queue

void makeNull() {

front = rear = -1;

itemCount = 0;

}

// returns the total elements fo the queue

int getSize() {

return itemCount;

}

};

void displayQueue(Queue &q) {

while(!q.isEmpty()) {

cout << q.peek() << ' ';

q.dequeue();

}

}

int main() {

Queue q;

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

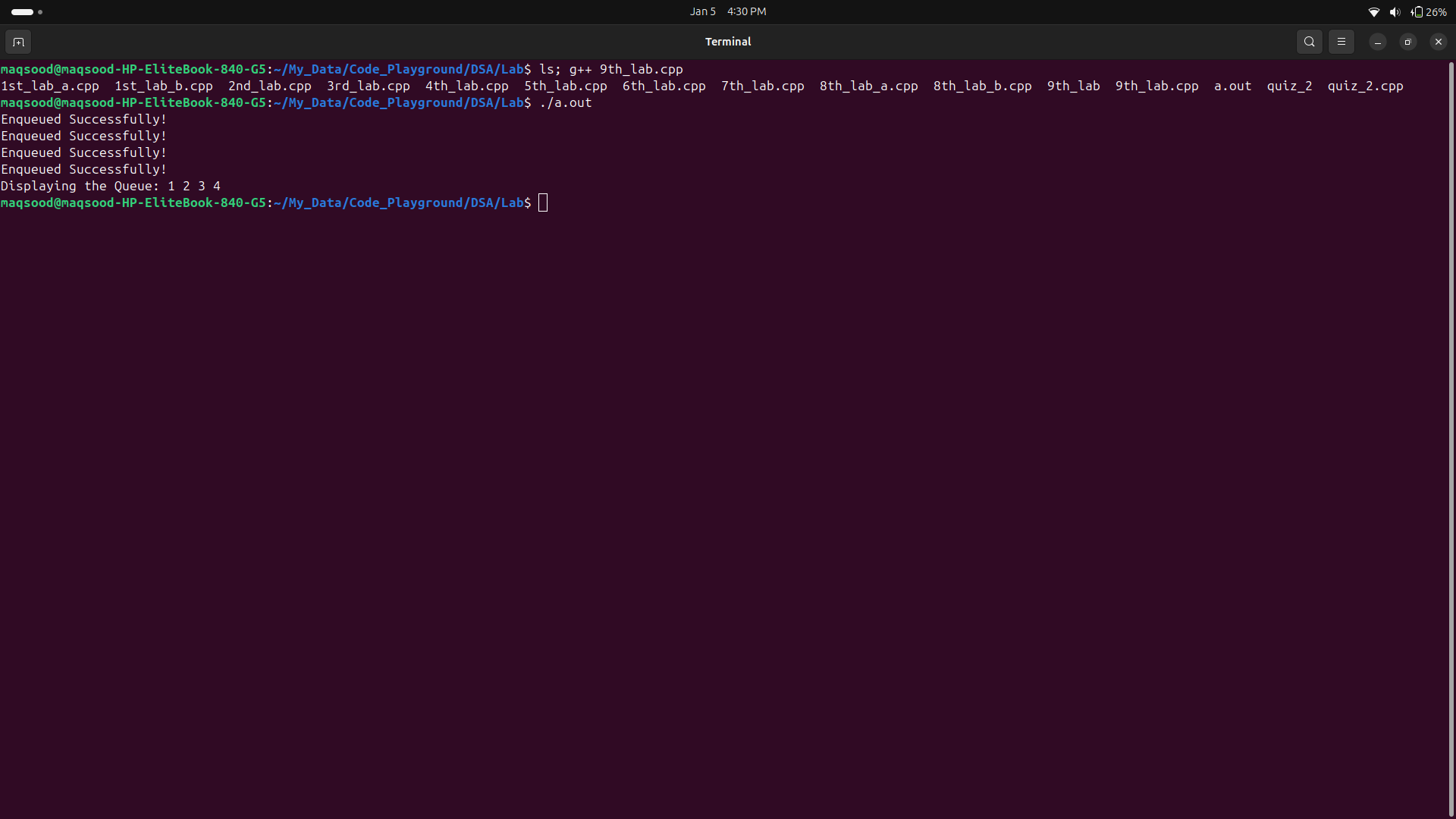
q.enqueue(4);

displayQueue(q);

return EXIT\_SUCCESS;

}

**OUTPUT:**



**Queue Data Structure Using Linked List:**

**Source Code:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int val) {

this->data = val;

this->next = NULL;

}

};

class Queue {

private:

Node\* front;

Node\* rear;

public:

Queue() : front(NULL), rear(NULL) {}

void enqueue(int val) {

Node\* temp = new Node(val);

if(!rear) {

rear = temp;

front = temp;

cout << "Enqueued Successfully!\n";

return;

}

rear->next = temp;

rear = temp; // Update the rear pointer

cout << "Enqueued Successfully!\n";

}

void dequeue() {

if(!front) {

cout << "Queue is empty!\n";

return;

}

Node\* temp = front;

front = front->next;

delete temp;

cout << "Dequeued successfully!\n";

}

int peek() {

if(!front) {

cout << "Queue is empty!\n";

exit(0);

}

return front->data;

}

bool isEmpty() {

return front == NULL;

}

// For displaying the queue without modifying it

void display() {

Node\* temp = front;

while(temp) {

cout << temp->data << ' ';

temp = temp->next;

}

cout << '\n';

}

};

int main(void) {

Queue q;

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

cout << "Queue elements: ";

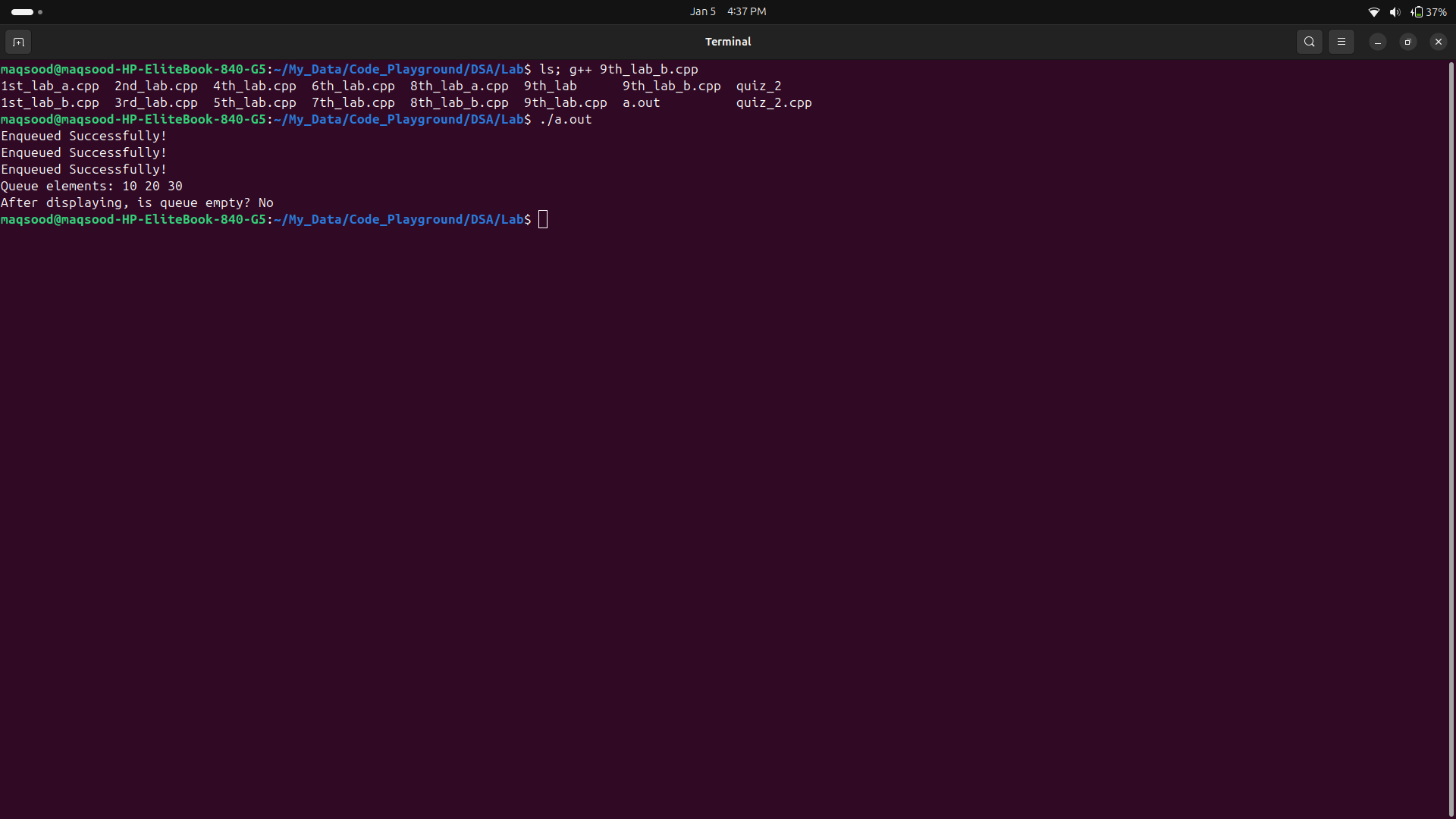
q.display(); // Display elements without dequeuing

cout << "After displaying, is queue empty? " << (q.isEmpty() ? "Yes" : "No") << '\n';

return EXIT\_SUCCESS;

}

**OUTPUT:**



**The End**