LAB TASK # 09

CODE # 1:

```
#include <iostream>
using namespace std;

class Node {
    public:
    int data;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* rear;

public:
    Queue() {
        front = rear = NULL;
    }

    bool isEmpty() {
        return front == NULL;
    }

    void enqueue(int value) {
        Node* newNode = new Node();
```

```
    newNode->data = value;
    newNode->next = NULL;

    if (isEmpty()) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    cout << value << " enqueued to queue\n";
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue is empty. Cannot dequeue.\n";
        return;
    }

    Node* temp = front;
    front = front->next;

    if (front == NULL) {
        rear = NULL;
    }

    cout << temp->data << " dequeued from queue\n";
    delete temp;
}
```

```
    int peek() {
        if (isEmpty()) {
            cout << "Queue is empty.\n";
            return -1;
        }
        return front->data;
    }
};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

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

    q.dequeue();
    q.dequeue();

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

    q.dequeue();
    q.dequeue();

    return 0;
```

```
}
```

CODE # 02:

```
#include <iostream>
using namespace std;

class Node {
    public:
    int data;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* rear;
    int count;

public:
    Queue() {
        front = rear = NULL;
        count = 0;
    }

    bool isEmpty() {
        return front == NULL;
    }
```

```
void enqueue(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = NULL;

    if (isEmpty()) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }

    count++;
    cout << value << " enqueued to queue\n";
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue is empty. Cannot dequeue.\n";
        return;
    }

    Node* temp = front;
    front = front->next;

    if (front == NULL) {
        rear = NULL;
    }
```

```
        cout << temp->data << " dequeued from queue\n";
        delete temp;

        count--;
    }

    int peek() {
        if (isEmpty()) {
            cout << "Queue is empty.\n";
            return -1;
        }
        return front->data;
    }

    int size() {
        return count;
    }
};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    cout << "Number of elements in the queue: " << q.size() << endl;
```

```
    q.dequeue();
    cout << "Number of elements in the queue: " << q.size() << endl;

    q.dequeue();
    cout << "Number of elements in the queue: " << q.size() << endl;

    q.dequeue();
    cout << "Number of elements in the queue: " << q.size() << endl;

    return 0;
}
```

CODE # 03:

```
#include <iostream>
using namespace std;

class Node {
    public:
    int data;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* rear;
    int count;
```

```
public:
    Queue() {
        front = rear = NULL;
        count = 0;
    }

    bool isEmpty() {
        return front == NULL;
    }

    void enqueue(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = NULL;

        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }

        count++;
        cout << value << " enqueued to queue\n";
    }

    int peek() {
        if (isEmpty()) {
```

```
            cout << "Queue is empty.\n";
            return -1;
        }
        return front->data;
    }

    int size() {
        return count;
    }

    void clear() {
        Node* current = front;
        while (current != NULL) {
            Node* nextNode = current->next;
            delete current;
            current = nextNode;
        }
        front = rear = NULL;
        count = 0;
        cout << "Queue cleared.\n";
    }
};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
```

```
    q.enqueue(30);

    cout << "Number of elements in the queue: " << q.size() << endl;

    q.clear();

    cout << "Number of elements in the queue after clear: " << q.size() << endl;

    return 0;
}
```