1.

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class LinkedList {

private:

Node\* head;

public:

LinkedList() : head(nullptr) {}

void insert(int value) {

Node\* newNode = new Node{value, head};

head = newNode;

}

void display() {

Node\* current = head;

while (current) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

void remove(int value) {

Node\* current = head;

Node\* prev = nullptr;

while (current && current->data != value) {

prev = current;

current = current->next;

}

if (current) {

if (prev) {

prev->next = current->next;

} else {

head = current->next;

}

delete current;

}

}

~LinkedList() {

while (head) {

remove(head->data);

}

}

};

int main() {

LinkedList list;

list.insert(1);

list.insert(2);

list.insert(3);

list.display();

list.remove(2);

list.display();

return 0;

}

2.

#include <iostream>

using namespace std;

void printTernary(int n) {

if (n == 0) {

cout << 0;

return;

}

string ternary = "";

while (n > 0) {

ternary = to\_string(n % 2) + ternary;

n /= 2;

}

cout << ternary << endl;

}

int main() {

int N;

cout << "输入一个非负整数N: ";

cin >> N;

printTernary(N);

return 0;

}

3.

#include <iostream>

#include <stack>

using namespace std;

bool isValidSequence(const string& sequence) {

stack<char> s;

for (char ch : sequence) {

if (ch == '(') {

s.push(ch);

} else if (ch == ')') {

if (s.empty() || s.top() != '(') return false;

s.pop();

}

}

return s.empty();

}

int main() {

string sequence;

cout << "输入数字序列: ";

cin >> sequence;

cout << (isValidSequence(sequence) ? "匹配正确" : "匹配错误") << endl;

return 0;

}

4.#include <iostream>

using namespace std;

class CircularQueue {

private:

int\* arr;

int front, rear, size;

public:

CircularQueue(int s) : size(s), front(0), rear(0) {

arr = new int[size];

}

void enqueue(int value) {

if ((rear + 1) % size == front) {

cout << "队列已满" << endl;

return;

}

arr[rear] = value;

rear = (rear + 1) % size;

}

void dequeue() {

if (front == rear) {

cout << "队列为空" << endl;

return;

}

front = (front + 1) % size;

}

void display() {

for (int i = front; i != rear; i = (i + 1) % size) {

cout << arr[i] << " ";

}

cout << endl;

}

~CircularQueue() {

delete[] arr;

}

};

int main() {

CircularQueue queue(5);

queue.enqueue(1);

queue.enqueue(2);

queue.enqueue(3);

queue.display();

queue.dequeue();

queue.display();

return 0;

}

5.

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class LinkedQueue {

private:

Node\* front;

Node\* rear;

public:

LinkedQueue() : front(nullptr), rear(nullptr) {}

void enqueue(int value) {

Node\* newNode = new Node{value, nullptr};

if (rear) {

rear->next = newNode;

} else {

front = newNode;

}

rear = newNode;

}

void dequeue() {

if (!front) {

cout << "队列为空" << endl;

return;

}

Node\* temp = front;

front = front->next;

if (!front) rear = nullptr;

delete temp;

}

void display() {

Node\* current = front;

while (current) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

~LinkedQueue() {

while (front) {

dequeue();

}

}

};

int main() {

LinkedQueue queue;

queue.enqueue(1);

queue.enqueue(2);

queue.enqueue(3);

queue.display();

queue.dequeue();

queue.display();

return 0;

}

6.

#include <iostream>

#include <stack>

using namespace std;

void reverseStack(stack<int>& s) {

stack<int> tempStack;

while (!s.empty()) {

tempStack.push(s.top());

s.pop();

}

s = tempStack;

}

int main() {

stack<int> s;

s.push(1);

s.push(2);

s.push(3);

reverseStack(s);

while (!s.empty()) {

cout << s.top() << " ";

s.pop();

}

cout << endl;

return 0;

}

7.

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

void reverseQueue(queue<int>& q) {

stack<int> s;

while (!q.empty()) {

s.push(q.front());

q.pop();

}

while (!s.empty()) {

q.push(s.top());

s.pop();

}

}

int main() {

queue<int> q;

q.push(1);

q.push(2);

q.push(3);

reverseQueue(q);

while (!q.empty()) {

cout << q.front() << " ";

q.pop();

}

cout << endl;

return 0;

}

8.

#include <iostream>

#include <string>

using namespace std;

void editLine() {

string line;

cout << "输入一行内容: ";

getline(cin, line);

for (char& ch : line) {

if (ch == '#') {

ch = '\0'; // 删除前一位

} else if (ch == '@') {

ch = ' '; // 将前面的内容清空

}

}

cout << "编辑后的内容: " << line << endl;

}

int main() {

editLine();

return 0;

}

9.

#include <iostream>

#include <string>

using namespace std;

bool isPalindrome(const string& str) {

int left = 0, right = str.size() - 1;

while (left < right) {

if (str[left] != str[right]) return false;

left++;

right--;

}

return true;

}

int main() {

string str;

cout << "输入字符串: ";

cin >> str;

cout << (isPalindrome(str) ? "是回文" : "不是回文") << endl;

return 0;

}