**Chương 5:**

Câu 1:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

// Hàm kiểm tra dấu ngoặc dư thừa

bool hasRedundantBrackets(const string &exp) {

    stack<char> s;

    for (char ch : exp) {

        if (ch == ')') {

            char top = s.top(); s.pop();

            bool hasOperator = false;

            while (!s.empty() && top != '(') {

                if (top == '+' || top == '-' || top == '\*' || top == '/') {

                    hasOperator = true;

                }

                top = s.top(); s.pop();

            }

            if (!hasOperator) {

                return true; // Dấu ngoặc dư thừa

            }

        } else {

            s.push(ch);

        }

    }

    return false;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Đọc bỏ dòng xuống dòng

    while (T--) {

        string exp;

        getline(cin, exp);

        if (hasRedundantBrackets(exp)) {

            cout << "Yes" << endl;

        } else {

            cout << "No" << endl;

        }

    }

    return 0;

}

3

((a+b))

Yes

(a + (b)/c)

Yes

(a + b\*(c-d))

No

Câu 2:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int minReversals(const string& expr) {

    int len = static\_cast<int>(expr.length());

    // Nếu độ dài lẻ, không thể cân bằng được

    if (len % 2 != 0) return -1;

    stack<char> s;

    // Duyệt từng ký tự trong chuỗi

    for (char ch : expr) {

        if (ch == ')' && !s.empty() && s.top() == '(') {

            s.pop();  // Nếu gặp "()", loại bỏ cặp này

        } else {

            s.push(ch);  // Đẩy vào stack nếu không thể khớp

        }

    }

    int unbalanced = static\_cast<int>(s.size());  // Tổng số dấu ngoặc chưa cân bằng

    int openCount = 0;  // Đếm số dấu '(' chưa khớp

    // Đếm số dấu '(' ở cuối stack

    while (!s.empty() && s.top() == '(') {

        s.pop();

        openCount++;

    }

    // Công thức tối ưu: (m+n)/2 + n%2

    return (unbalanced / 2 + openCount % 2);

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        string S;

        cin >> S;

        cout << minReversals(S) << endl;

    }

    return 0;

}

4

))((

2

((((

2

(((())

1

)(())(((

3

Câu 3:

#include <iostream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

string removeParentheses(string expr) {

    stack<int> s;

    string result;

    int n = expr.length();

    vector<bool> remove(n, false);

    for (int i = 0; i < n; ++i) {

        if (expr[i] == '(') {

            s.push(i);

        } else if (expr[i] == ')') {

            int j = s.top();

            s.pop();

            if (j > 0 && expr[j - 1] == '-') {

                for (int k = j + 1; k < i; ++k) {

                    if (expr[k] == '+') {

                        expr[k] = '-';

                    } else if (expr[k] == '-') {

                        expr[k] = '+';

                    }

                }

            }

            remove[j] = true;

            remove[i] = true;

        }

    }

    for (int i = 0; i < n; ++i) {

        if (!remove[i]) {

            result += expr[i];

        }

    }

    return result;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string P;

        getline(cin, P);

        cout << removeParentheses(P) << endl;

    }

    return 0;

}

2

a–(b+c)

a-b-c

a-(b-c-(d+e))-f

a-b+c+d+e-f

Câu 4:

#include <iostream>

#include <set>

#include <stack>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

void generateExpressions(string expr, int index, int open, int close, string current, set<string>& results) {

    if (index == expr.length()) {

        if (open == close) {

            results.insert(current);

        }

        return;

    }

    if (expr[index] == '(') {

        // Include the '('

        generateExpressions(expr, index + 1, open + 1, close, current + '(', results);

        // Exclude the '('

        generateExpressions(expr, index + 1, open, close, current, results);

    } else if (expr[index] == ')') {

        if (close < open) {

            // Include the ')'

            generateExpressions(expr, index + 1, open, close + 1, current + ')', results);

        }

        // Exclude the ')'

        generateExpressions(expr, index + 1, open, close, current, results);

    } else {

        // Include the current character

        generateExpressions(expr, index + 1, open, close, current + expr[index], results);

    }

}

bool isValidExpression(const string& expr) {

    stack<char> s;

    for (char ch : expr) {

        if (ch == '(') {

            s.push(ch);

        } else if (ch == ')') {

            if (s.empty() || s.top() != '(') {

                return false;

            }

            s.pop();

        }

    }

    return s.empty();

}

int main() {

    string expr;

    getline(cin, expr);

    set<string> results;

    generateExpressions(expr, 0, 0, 0, "", results);

    vector<string> validExpressions;

    for (const string& e : results) {

        if (isValidExpression(e)) {

            validExpressions.push\_back(e);

        }

    }

    sort(validExpressions.begin(), validExpressions.end());

    for (const string& e : validExpressions) {

        cout << e << endl;

    }

    return 0;

}

(1+(2\*(3+4)))

(1+(2\*(3+4)))

(1+(2\*3+4))

(1+2\*(3+4))

(1+2\*3+4)

1+(2\*(3+4))

1+(2\*3+4)

1+2\*(3+4)

1+2\*3+4

Câu 5:

#include <iostream>

#include <string>

using namespace std;

string normalizeExpression(const string& expr) {

    string result;

    for (char ch : expr) {

        if (ch != '(' && ch != ')') {

            result += ch;

        }

    }

    return result;

}

bool areExpressionsEqual(const string& P1, const string& P2) {

    return normalizeExpression(P1) == normalizeExpression(P2);

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string P1, P2;

        getline(cin, P1);

        getline(cin, P2);

        if (areExpressionsEqual(P1, P2)) {

            cout << "NO" << endl;

        } else {

            cout << "YES" << endl;

        }

    }

    return 0;

}

2

-(a+b+c)

-a-b-c

NO

a-b-(c-d)

a-b-c-d

YES

Câu 6:

#include <iostream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

int precedence(char op) {

    if (op == '+' || op == '-') return 1;

    if (op == '\*' || op == '/') return 2;

    if (op == '^') return 3;

    return 0;

}

string infixToPostfix(const string& exp) {

    stack<char> s;

    string result;

    for (char ch : exp) {

        if (isalnum(ch)) {

            result += ch;

        } else if (ch == '(') {

            s.push(ch);

        } else if (ch == ')') {

            while (!s.empty() && s.top() != '(') {

                result += s.top();

                s.pop();

            }

            s.pop(); // Pop the '('

        } else {

            while (!s.empty() && precedence(s.top()) >= precedence(ch)) {

                result += s.top();

                s.pop();

            }

            s.push(ch);

        }

    }

    while (!s.empty()) {

        result += s.top();

        s.pop();

    }

    return result;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << infixToPostfix(exp) << endl;

    }

    return 0;

}

2

(A+(B+C)

ABC++(

((A\*B)+C)

AB\*C+

Câu 7:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isOperator(char c) {

    return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');

}

string prefixToInfix(const string& exp) {

    stack<string> s;

    int length = exp.length();

    // Đọc biểu thức từ phải sang trái

    for (int i = length - 1; i >= 0; i--) {

        // Nếu ký tự là toán hạng, đẩy vào stack

        if (isalnum(exp[i])) {

            s.push(string(1, exp[i]));

        }

        // Nếu ký tự là toán tử

        else if (isOperator(exp[i])) {

            // Lấy hai toán hạng từ stack

            string op1 = s.top(); s.pop();

            string op2 = s.top(); s.pop();

            // Tạo biểu thức trung tố và đẩy lại vào stack

            string temp = "(" + op1 + exp[i] + op2 + ")";

            s.push(temp);

        }

    }

    // Phần tử cuối cùng trong stack là biểu thức trung tố

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << prefixToInfix(exp) << endl;

    }

    return 0;

}

2

\*+AB-CD

((A+B)\*(C-D))

\*-A/BC-/AKL

((A-(B/C))\*((A/K)-L))

Câu 8:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isOperator(char c) {

    return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');

}

string prefixToPostfix(const string& exp) {

    stack<string> s;

    int length = exp.length();

    // Đọc biểu thức từ phải sang trái

    for (int i = length - 1; i >= 0; i--) {

        // Nếu ký tự là toán hạng, đẩy vào stack

        if (isalnum(exp[i])) {

            s.push(string(1, exp[i]));

        }

        // Nếu ký tự là toán tử

        else if (isOperator(exp[i])) {

            // Lấy hai toán hạng từ stack

            string op1 = s.top(); s.pop();

            string op2 = s.top(); s.pop();

            // Tạo biểu thức hậu tố và đẩy lại vào stack

            string temp = op1 + op2 + exp[i];

            s.push(temp);

        }

    }

    // Phần tử cuối cùng trong stack là biểu thức hậu tố

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << prefixToPostfix(exp) << endl;

    }

    return 0;

}

2

\*+AB-CD

AB+CD-\*

\*-A/BC-/AKL

ABC/-AK/L-\*

Câu 9:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isOperator(char c) {

    return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');

}

string postfixToPrefix(const string& exp) {

    stack<string> s;

    int length = exp.length();

    // Đọc biểu thức từ trái sang phải

    for (int i = 0; i < length; i++) {

        // Nếu ký tự là toán hạng, đẩy vào stack

        if (isalnum(exp[i])) {

            s.push(string(1, exp[i]));

        }

        // Nếu ký tự là toán tử

        else if (isOperator(exp[i])) {

            // Lấy hai toán hạng từ stack

            string op2 = s.top(); s.pop();

            string op1 = s.top(); s.pop();

            // Tạo biểu thức tiền tố và đẩy lại vào stack

            string temp = exp[i] + op1 + op2;

            s.push(temp);

        }

    }

    // Phần tử cuối cùng trong stack là biểu thức tiền tố

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << postfixToPrefix(exp) << endl;

    }

    return 0;

}

2

AB+CD-\*

\*+AB-CD

ABC/-AK/L-\*

\*-A/BC-/AKL

Câu 10:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isOperator(char c) {

    return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');

}

string postfixToInfix(const string& exp) {

    stack<string> s;

    int length = exp.length();

    // Đọc biểu thức từ trái sang phải

    for (int i = 0; i < length; i++) {

        // Nếu ký tự là toán hạng, đẩy vào stack

        if (isalnum(exp[i])) {

            s.push(string(1, exp[i]));

        }

        // Nếu ký tự là toán tử

        else if (isOperator(exp[i])) {

            // Lấy hai toán hạng từ stack

            string op2 = s.top(); s.pop();

            string op1 = s.top(); s.pop();

            // Tạo biểu thức trung tố và đẩy lại vào stack

            string temp = "(" + op1 + exp[i] + op2 + ")";

            s.push(temp);

        }

    }

    // Phần tử cuối cùng trong stack là biểu thức trung tố

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << postfixToInfix(exp) << endl;

    }

    return 0;

}

2

ABC++

(A+(B+C))

AB\*C+

((A\*B)+C)

Câu 11:

#include <iostream>

#include <stack>

#include <string>

#include <cmath>

using namespace std;

int evaluatePostfix(const string& exp) {

    stack<int> s;

    int length = exp.length();

    for (int i = 0; i < length; i++) {

        // Nếu ký tự là toán hạng, đẩy vào stack

        if (isdigit(exp[i])) {

            s.push(exp[i] - '0');

        }

        // Nếu ký tự là toán tử

        else {

            int op2 = s.top(); s.pop();

            int op1 = s.top(); s.pop();

            int result;

            switch (exp[i]) {

                case '+': result = op1 + op2; break;

                case '-': result = op1 - op2; break;

                case '\*': result = op1 \* op2; break;

                case '/': result = op1 / op2; break;

                case '^': result = pow(op1, op2); break;

                default: result = 0; break;

            }

            s.push(result);

        }

    }

    // Phần tử cuối cùng trong stack là kết quả của biểu thức

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << evaluatePostfix(exp) << endl;

    }

    return 0;

}

2

231\*+9–

-4

875\*+9-

34

Câu 12:

#include <iostream>

#include <stack>

#include <string>

#include <cmath>

using namespace std;

int evaluatePrefix(const string& exp) {

    stack<int> s;

    int length = exp.length();

    // Đọc biểu thức từ phải sang trái

    for (int i = length - 1; i >= 0; i--) {

        // Nếu ký tự là toán hạng, đẩy vào stack

        if (isdigit(exp[i])) {

            s.push(exp[i] - '0');

        }

        // Nếu ký tự là toán tử

        else {

            int op1 = s.top(); s.pop();

            int op2 = s.top(); s.pop();

            int result;

            switch (exp[i]) {

                case '+': result = op1 + op2; break;

                case '-': result = op1 - op2; break;

                case '\*': result = op1 \* op2; break;

                case '/': result = op1 / op2; break;

                case '^': result = pow(op1, op2); break;

                default: result = 0; break;

            }

            s.push(result);

        }

    }

    // Phần tử cuối cùng trong stack là kết quả của biểu thức

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        cout << evaluatePrefix(exp) << endl;

    }

    return 0;

}

2

-+8/632

8

-+7\*45+20

25

Câu 13:

#include <iostream>

#include <stack>

#include <string>

#include <sstream>

#include <cctype>

#include <cmath>

using namespace std;

int precedence(char op) {

    if (op == '+' || op == '-') return 1;

    if (op == '\*' || op == '/') return 2;

    return 0;

}

string infixToPostfix(const string& exp) {

    stack<char> s;

    string result;

    stringstream ss(exp);

    string token;

    while (ss >> token) {

        if (isdigit(token[0])) {

            result += token + ' ';

        } else if (token == "(") {

            s.push('(');

        } else if (token == ")") {

            while (!s.empty() && s.top() != '(') {

                result += s.top();

                result += ' ';

                s.pop();

            }

            s.pop(); // Pop the '('

        } else {

            while (!s.empty() && precedence(s.top()) >= precedence(token[0])) {

                result += s.top();

                result += ' ';

                s.pop();

            }

            s.push(token[0]);

        }

    }

    while (!s.empty()) {

        result += s.top();

        result += ' ';

        s.pop();

    }

    return result;

}

int evaluatePostfix(const string& exp) {

    stack<int> s;

    stringstream ss(exp);

    string token;

    while (ss >> token) {

        if (isdigit(token[0])) {

            s.push(stoi(token));

        } else {

            int op2 = s.top(); s.pop();

            int op1 = s.top(); s.pop();

            int result;

            switch (token[0]) {

                case '+': result = op1 + op2; break;

                case '-': result = op1 - op2; break;

                case '\*': result = op1 \* op2; break;

                case '/': result = op1 / op2; break;

                default: result = 0; break;

            }

            s.push(result);

        }

    }

    return s.top();

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string exp;

        getline(cin, exp);

        string postfix = infixToPostfix(exp);

        cout << evaluatePostfix(postfix) << endl;

    }

    return 0;

}

4

6\*3+2-(6-4/2)

6

100+99\*22

100

6\*((4\*3)+5)

6

1-2

1

Câu 14:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

string findSmallestNumber(const string& S) {

    stack<int> st;

    string result;

    int num = 1;

    for (char ch : S) {

        st.push(num++);

        if (ch == 'I') {

            while (!st.empty()) {

                result += to\_string(st.top());

                st.pop();

            }

        }

    }

    st.push(num++);

    while (!st.empty()) {

        result += to\_string(st.top());

        st.pop();

    }

    return result;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string S;

        getline(cin, S);

        cout << findSmallestNumber(S) << endl;

    }

    return 0;

}

4

I

12

d

21

dd

321

DDIDDIID

321654798

Câu 15:

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

vector<int> findNextGreaterElements(const vector<int>& arr) {

    int n = arr.size();

    vector<int> nextGreater(n, -1);

    stack<int> s;

    for (int i = n - 1; i >= 0; --i) {

        while (!s.empty() && s.top() <= arr[i]) {

            s.pop();

        }

        if (!s.empty()) {

            nextGreater[i] = s.top();

        }

        s.push(arr[i]);

    }

    return nextGreater;

}

vector<int> findNextSmallerElements(const vector<int>& arr) {

    int n = arr.size();

    vector<int> nextSmaller(n, -1);

    stack<int> s;

    for (int i = n - 1; i >= 0; --i) {

        while (!s.empty() && s.top() >= arr[i]) {

            s.pop();

        }

        if (!s.empty()) {

            nextSmaller[i] = s.top();

        }

        s.push(arr[i]);

    }

    return nextSmaller;

}

vector<int> findResult(const vector<int>& arr) {

    vector<int> nextGreater = findNextGreaterElements(arr);

    vector<int> result(arr.size(), -1);

    for (int i = 0; i < arr.size(); ++i) {

        if (nextGreater[i] != -1) {

            vector<int> subArr(arr.begin() + i + 1, arr.end());

            vector<int> nextSmaller = findNextSmallerElements(subArr);

            for (int j = 0; j < subArr.size(); ++j) {

                if (subArr[j] == nextGreater[i]) {

                    result[i] = nextSmaller[j];

                    break;

                }

            }

        }

    }

    return result;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int n;

        cin >> n;

        vector<int> arr(n);

        for (int i = 0; i < n; ++i) {

            cin >> arr[i];

        }

        vector<int> result = findResult(arr);

        for (int val : result) {

            cout << val << " ";

        }

        cout << endl;

    }

    return 0;

}

2

7

5 1 9 2 5 1 7

2 2 -1 1 -1 -1 -1

8

4 8 2 1 9 5 6 3

2 5 5 5 -1 3 -1 -1

Câu 16:

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

long long largestRectangleArea(const vector<int>& heights) {

    int n = heights.size();

    stack<int> s;

    long long maxArea = 0;

    int i = 0;

    while (i < n) {

        if (s.empty() || heights[s.top()] <= heights[i]) {

            s.push(i++);

        } else {

            int h = heights[s.top()];

            s.pop();

            long long width = s.empty() ? i : i - s.top() - 1;

            maxArea = max(maxArea, h \* width);

        }

    }

    while (!s.empty()) {

        int h = heights[s.top()];

        s.pop();

        long long width = s.empty() ? i : i - s.top() - 1;

        maxArea = max(maxArea, h \* width);

    }

    return maxArea;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        vector<int> heights(N);

        for (int i = 0; i < N; ++i) {

            cin >> heights[i];

        }

        cout << largestRectangleArea(heights) << endl;

    }

    return 0;

}

2

7

6 2 5 4 5 1 6

12

3

2 2 2

6

Câu 17:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int longestValidParentheses(const string& s) {

    stack<int> st;

    int maxLength = 0;

    st.push(-1); // Khởi tạo stack với giá trị -1 để tính toán độ dài

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

        if (s[i] == '(') {

            st.push(i);

        } else {

            st.pop();

            if (st.empty()) {

                st.push(i);

            } else {

                maxLength = max(maxLength, i - st.top());

            }

        }

    }

    return maxLength;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string S;

        getline(cin, S);

        cout << longestValidParentheses(S) << endl;

    }

    return 0;

}

3

((()

2

)()())

4

()(()))))

6

Câu 18:

#include <iostream>

#include <vector>

#include <stack>

#include <unordered\_map>

using namespace std;

vector<int> findNextGreaterFrequency(const vector<int>& arr) {

    int n = arr.size();

    vector<int> result(n, -1);

    unordered\_map<int, int> freq;

    stack<int> s;

    // Đếm số lần xuất hiện của mỗi phần tử

    for (int num : arr) {

        freq[num]++;

    }

    // Duyệt qua mảng từ phải sang trái

    for (int i = n - 1; i >= 0; --i) {

        while (!s.empty() && freq[arr[s.top()]] <= freq[arr[i]]) {

            s.pop();

        }

        if (!s.empty()) {

            result[i] = arr[s.top()];

        }

        s.push(i);

    }

    return result;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int n;

        cin >> n;

        vector<int> arr(n);

        for (int i = 0; i < n; ++i) {

            cin >> arr[i];

        }

        vector<int> result = findNextGreaterFrequency(arr);

        for (int val : result) {

            cout << val << " ";

        }

        cout << endl;

    }

    return 0;

}

1

7

1 1 2 3 4 2 1

-1 -1 1 2 2 1 -1

Câu 19:

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

vector<int> calculateStockSpan(const vector<int>& prices) {

    int n = prices.size();

    vector<int> span(n, 1);

    stack<int> s;

    for (int i = 0; i < n; ++i) {

        while (!s.empty() && prices[s.top()] <= prices[i]) {

            s.pop();

        }

        span[i] = s.empty() ? (i + 1) : (i - s.top());

        s.push(i);

    }

    return span;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        vector<int> prices(N);

        for (int i = 0; i < N; ++i) {

            cin >> prices[i];

        }

        vector<int> result = calculateStockSpan(prices);

        for (int val : result) {

            cout << val << " ";

        }

        cout << endl;

    }

    return 0;

}

1

7

100 80 60 70 60 75 85

1 1 1 2 1 4 6

Câu 20:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

string decodeString(const string& s) {

    stack<string> strStack;

    stack<int> numStack;

    string currentStr;

    int currentNum = 0;

    for (char ch : s) {

        if (isdigit(ch)) {

            currentNum = currentNum \* 10 + (ch - '0');

        } else if (ch == '[') {

            numStack.push(currentNum);

            strStack.push(currentStr);

            currentNum = 0;

            currentStr.clear();

        } else if (ch == ']') {

            string temp = currentStr;

            currentStr = strStack.top();

            strStack.pop();

            int repeatTimes = numStack.top();

            numStack.pop();

            while (repeatTimes--) {

                currentStr += temp;

            }

        } else {

            currentStr += ch;

        }

    }

    return currentStr;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string str;

        getline(cin, str);

        cout << decodeString(str) << endl;

    }

    return 0;

}

2

1[b]

b

3[b2[ca]]

bcacabcacabcaca

Câu 21:

#include <iostream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

int longestValidParenthesesSum(const string& s) {

    int n = s.length();

    stack<int> st;

    vector<bool> valid(n, false);

    // Sử dụng stack để đánh dấu các vị trí của các ký tự ngoặc đúng

    for (int i = 0; i < n; ++i) {

        if (s[i] == '(') {

            st.push(i);

        } else if (!st.empty()) {

            valid[i] = true;

            valid[st.top()] = true;

            st.pop();

        }

    }

    // Tính tổng độ dài của các biểu thức con viết đúng

    int maxLength = 0, currentLength = 0;

    for (int i = 0; i < n; ++i) {

        if (valid[i]) {

            currentLength++;

        } else {

            maxLength += (currentLength / 2) \* 2;

            currentLength = 0;

        }

    }

    maxLength += (currentLength / 2) \* 2;

    return maxLength;

}

int main() {

    int T;

    cin >> T;

    cin.ignore(); // Bỏ qua ký tự newline sau số lượng bộ test

    while (T--) {

        string P;

        getline(cin, P);

        cout << longestValidParenthesesSum(P) << endl;

    }

    return 0;

}

4

(()(

2

()()((

4

((()()())))

10

()(())(

6

**Chương 6:**

Câu 1:

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

vector<string> generateBinaryNumbers(int n) {

    vector<string> result;

    queue<string> q;

    q.push("1");

    while (n--) {

        string s1 = q.front();

        q.pop();

        result.push\_back(s1);

        string s2 = s1;

        q.push(s1 + "0");

        q.push(s2 + "1");

    }

    return result;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int n;

        cin >> n;

        vector<string> binaryNumbers = generateBinaryNumbers(n);

        for (const string& num : binaryNumbers) {

            cout << num << " ";

        }

        cout << endl;

    }

    return 0;

}

2

2

1 10

5

1 10 11 100 101

Câu 2:

#include <iostream>

#include <queue>

#include <string>

using namespace std;

string findSmallestMultiple(int N) {

    queue<string> q;

    q.push("9");

    while (!q.empty()) {

        string num = q.front();

        q.pop();

        // Chuyển đổi chuỗi thành số nguyên và kiểm tra xem nó có chia hết cho N không

        long long number = stoll(num);

        if (number % N == 0) {

            return num;

        }

        // Thêm các số mới vào hàng đợi

        q.push(num + "0");

        q.push(num + "9");

    }

    return "";

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        cout << findSmallestMultiple(N) << endl;

    }

    return 0;

}

2

5

90

7

9009

Câu 3:

#include <iostream>

#include <queue>

#include <string>

using namespace std;

int countBDN(long long N) {

    queue<string> q;

    q.push("1");

    int count = 0;

    while (!q.empty()) {

        string num = q.front();

        q.pop();

        // Chuyển đổi chuỗi thành số nguyên và kiểm tra xem nó có nhỏ hơn N không

        long long number = stoll(num);

        if (number >= N) {

            break;

        }

        count++;

        // Thêm các số mới vào hàng đợi

        q.push(num + "0");

        q.push(num + "1");

    }

    return count;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        long long N;

        cin >> N;

        cout << countBDN(N) << endl;

    }

    return 0;

}

3

10

1

100

3

200

7

Câu 4:

#include <iostream>

#include <queue>

#include <string>

using namespace std;

string findSmallestBDN(int N) {

    queue<string> q;

    q.push("1");

    while (!q.empty()) {

        string num = q.front();

        q.pop();

        // Chuyển đổi chuỗi thành số nguyên và kiểm tra xem nó có chia hết cho N không

        long long number = stoll(num);

        if (number % N == 0) {

            return num;

        }

        // Thêm các số mới vào hàng đợi

        q.push(num + "0");

        q.push(num + "1");

    }

    return "";

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        cout << findSmallestBDN(N) << endl;

    }

    return 0;

}

3

2

10

12

11100

17

11101

Câu 5:

#include <iostream>

#include <queue>

#include <unordered\_set>

using namespace std;

int minOperations(int S, int T) {

    if (S >= T) {

        return S - T;

    }

    queue<pair<int, int>> q;

    unordered\_set<int> visited;

    q.push({S, 0});

    visited.insert(S);

    while (!q.empty()) {

        int current = q.front().first;

        int steps = q.front().second;

        q.pop();

        if (current == T) {

            return steps;

        }

        if (current \* 2 == T || current - 1 == T) {

            return steps + 1;

        }

        if (current \* 2 < T \* 2 && visited.find(current \* 2) == visited.end()) {

            q.push({current \* 2, steps + 1});

            visited.insert(current \* 2);

        }

        if (current - 1 > 0 && visited.find(current - 1) == visited.end()) {

            q.push({current - 1, steps + 1});

            visited.insert(current - 1);

        }

    }

    return -1; // Không tìm thấy đường đi

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int S, T;

        cin >> S >> T;

        cout << minOperations(S, T) << endl;

    }

    return 0;

}

3

2 5

4

3 7

4

7 4

3

Câu 6:

#include <iostream>

#include <queue>

#include <unordered\_set>

using namespace std;

int minOperations(int N) {

    if (N == 1) {

        return 0;

    }

    queue<pair<int, int>> q;

    unordered\_set<int> visited;

    q.push({N, 0});

    visited.insert(N);

    while (!q.empty()) {

        int current = q.front().first;

        int steps = q.front().second;

        q.pop();

        // Thao tác (a): Trừ N đi 1

        if (current - 1 == 1) {

            return steps + 1;

        }

        if (visited.find(current - 1) == visited.end()) {

            q.push({current - 1, steps + 1});

            visited.insert(current - 1);

        }

        // Thao tác (b): N = max(u, v) nếu u \* v = N

        for (int i = 2; i \* i <= current; ++i) {

            if (current % i == 0) {

                int maxFactor = max(i, current / i);

                if (maxFactor == 1) {

                    return steps + 1;

                }

                if (visited.find(maxFactor) == visited.end()) {

                    q.push({maxFactor, steps + 1});

                    visited.insert(maxFactor);

                }

            }

        }

    }

    return -1; // Không tìm thấy đường đi

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        cout << minOperations(N) << endl;

    }

    return 0;

}

3

17

4

50

5

100

5

Câu 7:

#include <iostream>

#include <queue>

#include <unordered\_set>

#include <vector>

#include <string>

using namespace std;

// Hàm kiểm tra xem hai xâu có khác nhau duy nhất một ký tự hay không

bool isAdjacent(const string& a, const string& b) {

    int count = 0;

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

        if (a[i] != b[i]) {

            count++;

        }

        if (count > 1) {

            return false;

        }

    }

    return count == 1;

}

int shortestPathLength(const vector<string>& words, const string& start, const string& end) {

    unordered\_set<string> wordSet(words.begin(), words.end());

    if (wordSet.find(end) == wordSet.end()) {

        return -1;

    }

    queue<pair<string, int>> q;

    q.push({start, 1});

    wordSet.erase(start);

    while (!q.empty()) {

        string currentWord = q.front().first;

        int currentLength = q.front().second;

        q.pop();

        for (auto it = wordSet.begin(); it != wordSet.end();) {

            if (isAdjacent(currentWord, \*it)) {

                if (\*it == end) {

                    return currentLength + 1;

                }

                q.push({\*it, currentLength + 1});

                it = wordSet.erase(it);

            } else {

                ++it;

            }

        }

    }

    return -1;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int n;

        string s, t;

        cin >> n >> s >> t;

        vector<string> words(n);

        for (int i = 0; i < n; ++i) {

            cin >> words[i];

        }

        int result = shortestPathLength(words, s, t);

        cout << result << endl;

    }

    return 0;

}

1

8 TOON PLEA

POON TOON PLEE SAME POIE PLEA PLIE POIN

7

Câu 8:

#include <iostream>

#include <vector>

#include <unordered\_set>

using namespace std;

bool isValid(int num) {

    unordered\_set<int> digits;

    while (num > 0) {

        int digit = num % 10;

        if (digit > 5 || digits.find(digit) != digits.end()) {

            return false;

        }

        digits.insert(digit);

        num /= 10;

    }

    return true;

}

int countValidNumbers(int L, int R) {

    int count = 0;

    for (int i = L; i <= R; ++i) {

        if (isValid(i)) {

            count++;

        }

    }

    return count;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int L, R;

        cin >> L >> R;

        cout << countValidNumbers(L, R) << endl;

    }

    return 0;

}

2

4 13

5

100 1000

100

Câu 9:

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

struct Point {

    int x, y, steps;

};

bool isValid(int x, int y, int N, const vector<vector<char>>& board, vector<vector<bool>>& visited) {

    return (x >= 0 && x < N && y >= 0 && y < N && board[x][y] == '.' && !visited[x][y]);

}

int minSteps(int N, const vector<vector<char>>& board, int a, int b, int c, int d) {

    vector<vector<bool>> visited(N, vector<bool>(N, false));

    queue<Point> q;

    q.push({a, b, 0});

    visited[a][b] = true;

    int directions[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

    while (!q.empty()) {

        Point current = q.front();

        q.pop();

        if (current.x == c && current.y == d) {

            return current.steps;

        }

        for (int i = 0; i < 4; ++i) {

            int nx = current.x + directions[i][0];

            int ny = current.y + directions[i][1];

            while (isValid(nx, ny, N, board, visited)) {

                visited[nx][ny] = true;

                q.push({nx, ny, current.steps + 1});

                nx += directions[i][0];

                ny += directions[i][1];

            }

        }

    }

    return -1; // Không tìm thấy đường đi

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        vector<vector<char>> board(N, vector<char>(N));

        for (int i = 0; i < N; ++i) {

            for (int j = 0; j < N; ++j) {

                cin >> board[i][j];

            }

        }

        int a, b, c, d;

        cin >> a >> b >> c >> d;

        cout << minSteps(N, board, a, b, c, d) << endl;

    }

    return 0;

}

1

3

.X.

.X.

...

0 0 0 2

3

Câu 10:

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

struct Point {

    int x, y, days;

};

bool isValid(int x, int y, int R, int C, const vector<vector<int>>& grid, vector<vector<bool>>& visited) {

    return (x >= 0 && x < R && y >= 0 && y < C && grid[x][y] == 1 && !visited[x][y]);

}

int minDaysToGerminate(int R, int C, vector<vector<int>>& grid) {

    queue<Point> q;

    vector<vector<bool>> visited(R, vector<bool>(C, false));

    int totalSeeds = 0;

    int germinatedSeeds = 0;

    // Đẩy tất cả các cây non vào hàng đợi và đánh dấu chúng là đã thăm

    for (int i = 0; i < R; ++i) {

        for (int j = 0; j < C; ++j) {

            if (grid[i][j] == 2) {

                q.push({i, j, 0});

                visited[i][j] = true;

            }

            if (grid[i][j] == 1) {

                totalSeeds++;

            }

        }

    }

    int directions[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

    int days = 0;

    while (!q.empty()) {

        Point current = q.front();

        q.pop();

        for (int i = 0; i < 4; ++i) {

            int nx = current.x + directions[i][0];

            int ny = current.y + directions[i][1];

            if (isValid(nx, ny, R, C, grid, visited)) {

                visited[nx][ny] = true;

                q.push({nx, ny, current.days + 1});

                germinatedSeeds++;

                days = current.days + 1;

            }

        }

    }

    return (germinatedSeeds == totalSeeds) ? days : -1;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int R, C;

        cin >> R >> C;

        vector<vector<int>> grid(R, vector<int>(C));

        for (int i = 0; i < R; ++i) {

            for (int j = 0; j < C; ++j) {

                cin >> grid[i][j];

            }

        }

        cout << minDaysToGerminate(R, C, grid) << endl;

    }

    return 0;

}

2

3 5

2 1 0 2 1

1 0 1 2 1

1 0 0 2 1

2

3 5

2 1 0 2 1

0 0 1 2 1

1 0 0 2 1

-1

Câu 11:

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

struct Point {

    int x, y, z, steps;

};

bool isValid(int x, int y, int z, int A, int B, int C, const vector<vector<vector<char>>>& grid, vector<vector<vector<bool>>>& visited) {

    return (x >= 0 && x < A && y >= 0 && y < B && z >= 0 && z < C && grid[x][y][z] != '#' && !visited[x][y][z]);

}

int minSteps(int A, int B, int C, const vector<vector<vector<char>>>& grid, Point start, Point end) {

    vector<vector<vector<bool>>> visited(A, vector<vector<bool>>(B, vector<bool>(C, false)));

    queue<Point> q;

    q.push(start);

    visited[start.x][start.y][start.z] = true;

    int directions[6][3] = {{0, 0, 1}, {0, 0, -1}, {0, 1, 0}, {0, -1, 0}, {1, 0, 0}, {-1, 0, 0}};

    while (!q.empty()) {

        Point current = q.front();

        q.pop();

        if (current.x == end.x && current.y == end.y && current.z == end.z) {

            return current.steps;

        }

        for (int i = 0; i < 6; ++i) {

            int nx = current.x + directions[i][0];

            int ny = current.y + directions[i][1];

            int nz = current.z + directions[i][2];

            if (isValid(nx, ny, nz, A, B, C, grid, visited)) {

                visited[nx][ny][nz] = true;

                q.push({nx, ny, nz, current.steps + 1});

            }

        }

    }

    return -1; // Không tìm thấy đường đi

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int A, B, C;

        cin >> A >> B >> C;

        vector<vector<vector<char>>> grid(A, vector<vector<char>>(B, vector<char>(C)));

        Point start, end;

        for (int i = 0; i < A; ++i) {

            for (int j = 0; j < B; ++j) {

                for (int k = 0; k < C; ++k) {

                    cin >> grid[i][j][k];

                    if (grid[i][j][k] == 'S') {

                        start = {i, j, k, 0};

                    } else if (grid[i][j][k] == 'E') {

                        end = {i, j, k, 0};

                    }

                }

            }

        }

        cout << minSteps(A, B, C, grid, start, end) << endl;

    }

    return 0;

}

2

3 4 5

S....

.###.

.##..

###.#

#####

#####

##.##

##...

#####

#####

#.###

####E

11

1 3 3

S##

#E#

###

-1

Câu 12:

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

vector<string> generateLocPhatNumbers(int N) {

    vector<string> result;

    queue<string> q;

    q.push("6");

    q.push("8");

    while (!q.empty()) {

        string num = q.front();

        q.pop();

        if (num.length() > N) {

            break;

        }

        result.push\_back(num);

        q.push(num + "6");

        q.push(num + "8");

    }

    sort(result.rbegin(), result.rend()); // Sắp xếp theo thứ tự giảm dần

    return result;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        vector<string> locPhatNumbers = generateLocPhatNumbers(N);

        for (const string& num : locPhatNumbers) {

            cout << num << " ";

        }

        cout << endl;

    }

    return 0;

}

2

2

88 86 8 68 66 6

3

888 886 88 868 866 86 8 688 686 68 668 666 66 6

Câu 13:

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

vector<string> generateLocPhatNumbers(int N) {

    vector<string> result;

    queue<string> q;

    q.push("6");

    q.push("8");

    while (!q.empty()) {

        string num = q.front();

        q.pop();

        if (num.length() > N) {

            break;

        }

        result.push\_back(num);

        q.push(num + "6");

        q.push(num + "8");

    }

    sort(result.begin(), result.end()); // Sắp xếp theo thứ tự tăng dần

    return result;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        vector<string> locPhatNumbers = generateLocPhatNumbers(N);

        cout << locPhatNumbers.size() << endl;

        for (const string& num : locPhatNumbers) {

            cout << num << " ";

        }

        cout << endl;

    }

    return 0;

}

2

2

6

6 66 68 8 86 88

3

14

6 66 666 668 68 686 688 8 86 866 868 88 886 888

Câu 14:

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

vector<string> generateLocPhatNumbers(int N) {

    vector<string> result;

    queue<string> q;

    q.push("6");

    q.push("8");

    while (!q.empty()) {

        string num = q.front();

        q.pop();

        if (num.length() > N) {

            break;

        }

        result.push\_back(num);

        q.push(num + "6");

        q.push(num + "8");

    }

    sort(result.rbegin(), result.rend()); // Sắp xếp theo thứ tự giảm dần

    return result;

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int N;

        cin >> N;

        vector<string> locPhatNumbers = generateLocPhatNumbers(N);

        cout << locPhatNumbers.size() << endl;

        for (const string& num : locPhatNumbers) {

            cout << num << " ";

        }

        cout << endl;

    }

    return 0;

}

2

2

6

88 86 8 68 66 6

3

14

888 886 88 868 866 86 8 688 686 68 668 666 66 6

Câu 15:

#include <iostream>

#include <queue>

#include <vector>

#include <unordered\_set>

#include <string>

using namespace std;

// Hàm kiểm tra xem một số có phải là số nguyên tố hay không

bool isPrime(int num) {

    if (num < 2) return false;

    for (int i = 2; i \* i <= num; ++i) {

        if (num % i == 0) return false;

    }

    return true;

}

// Hàm tìm số bước dịch chuyển ít nhất từ S đến T

int minTransformations(int S, int T) {

    if (S == T) return 0;

    queue<pair<int, int>> q;

    unordered\_set<int> visited;

    q.push({S, 0});

    visited.insert(S);

    while (!q.empty()) {

        int current = q.front().first;

        int steps = q.front().second;

        q.pop();

        string currentStr = to\_string(current);

        for (int i = 0; i < 4; ++i) {

            char originalDigit = currentStr[i];

            for (char digit = '0'; digit <= '9'; ++digit) {

                if (digit == originalDigit) continue;

                currentStr[i] = digit;

                int nextNum = stoi(currentStr);

                if (nextNum == T) return steps + 1;

                if (nextNum >= 1000 && isPrime(nextNum) && visited.find(nextNum) == visited.end()) {

                    q.push({nextNum, steps + 1});

                    visited.insert(nextNum);

                }

            }

            currentStr[i] = originalDigit;

        }

    }

    return -1; // Không tìm thấy đường đi

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int S, T;

        cin >> S >> T;

        cout << minTransformations(S, T) << endl;

    }

    return 0;

}

2

1033 8179

6

1033 8779

5

Câu 16:

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

struct Point {

    int x, y, steps;

};

bool isValid(int x, int y, int M, int N, const vector<vector<int>>& A, vector<vector<bool>>& visited) {

    return (x >= 0 && x < M && y >= 0 && y < N && !visited[x][y]);

}

int minSteps(int M, int N, const vector<vector<int>>& A) {

    vector<vector<bool>> visited(M, vector<bool>(N, false));

    queue<Point> q;

    q.push({0, 0, 0});

    visited[0][0] = true;

    while (!q.empty()) {

        Point current = q.front();

        q.pop();

        if (current.x == M - 1 && current.y == N - 1) {

            return current.steps;

        }

        int jump = A[current.x][current.y];

        // Di chuyển đến vị trí A[i][j + A[i][j]]

        int newY = current.y + jump;

        if (isValid(current.x, newY, M, N, A, visited)) {

            visited[current.x][newY] = true;

            q.push({current.x, newY, current.steps + 1});

        }

        // Di chuyển đến vị trí A[i + A[i][j]][j]

        int newX = current.x + jump;

        if (isValid(newX, current.y, M, N, A, visited)) {

            visited[newX][current.y] = true;

            q.push({newX, current.y, current.steps + 1});

        }

    }

    return -1; // Không tìm thấy đường đi

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        int M, N;

        cin >> M >> N;

        vector<vector<int>> A(M, vector<int>(N));

        for (int i = 0; i < M; ++i) {

            for (int j = 0; j < N; ++j) {

                cin >> A[i][j];

            }

        }

        cout << minSteps(M, N, A) << endl;

    }

    return 0;

}

1

3 3

2 1 2

1 1 1

1 1 1

2

Câu 17:

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

// Các hướng di chuyển của quân mã

int dx[8] = {1, 1, -1, -1, 2, 2, -2, -2};

int dy[8] = {2, -2, 2, -2, 1, -1, 1, -1};

// Hàm chuyển đổi tọa độ từ ký tự sang số nguyên

pair<int, int> toCoords(string pos) {

    return {pos[0] - 'a', pos[1] - '1'};

}

// Thuật toán BFS tìm số bước ít nhất

int bfs(string start, string end) {

    if (start == end) return 0;

    queue<pair<int, int>> q;

    unordered\_map<int, unordered\_map<int, int>> dist;

    auto [sx, sy] = toCoords(start);

    auto [ex, ey] = toCoords(end);

    q.push({sx, sy});

    dist[sx][sy] = 0;

    while (!q.empty()) {

        auto [x, y] = q.front();

        q.pop();

        for (int i = 0; i < 8; i++) {

            int nx = x + dx[i];

            int ny = y + dy[i];

            if (nx >= 0 && nx < 8 && ny >= 0 && ny < 8 && dist.find(nx) == dist.end() || dist[nx].find(ny) == dist[nx].end()) {

                dist[nx][ny] = dist[x][y] + 1;

                if (nx == ex && ny == ey) return dist[nx][ny];

                q.push({nx, ny});

            }

        }

    }

    return -1; // Không thể xảy ra

}

int main() {

    int T;

    cin >> T;

    while (T--) {

        string start, end;

        cin >> start >> end;

        cout << bfs(start, end) << endl;

    }

    return 0;

}

8

e2 e4

2

a1 b2

2

b2 c3

2

a1 h8

6

a1 h7

5

h8 a1

6

b1 c3

1

f6 f6

0

Câu 18:

#include <iostream>

#include <queue>

#include <unordered\_set>

#include <vector>

using namespace std;

struct State {

    vector<int> board;

    int steps;

    State(vector<int> b, int s) : board(b), steps(s) {}

};

vector<int> rotateLeft(const vector<int>& state) {

    return {state[3], state[0], state[2], state[4], state[1], state[5]};

}

vector<int> rotateRight(const vector<int>& state) {

    return {state[0], state[4], state[1], state[3], state[5], state[2]};

}

int minMoves(vector<int> start, vector<int> target) {

    if (start == target) return 0;

    queue<State> q;

    unordered\_set<string> visited;

    auto vecToStr = [](const vector<int>& v) {

        string res;

        for (int num : v) res += to\_string(num) + ",";

        return res;

    };

    q.push(State(start, 0));

    visited.insert(vecToStr(start));

    while (!q.empty()) {

        State current = q.front(); q.pop();

        vector<int> leftRot = rotateLeft(current.board);

        vector<int> rightRot = rotateRight(current.board);

        for (auto& newState : {leftRot, rightRot}) {

            if (newState == target) return current.steps + 1;

            string newStr = vecToStr(newState);

            if (visited.find(newStr) == visited.end()) {

                visited.insert(newStr);

                q.push(State(newState, current.steps + 1));

            }

        }

    }

    return -1; // Không thể đạt được trạng thái đích (không xảy ra)

}

int main() {

    int t;

    cin >> t;

    while (t--) {

        vector<int> start(6), target(6);

        for (int i = 0; i < 6; ++i) cin >> start[i];

        for (int i = 0; i < 6; ++i) cin >> target[i];

        cout << minMoves(start, target) << endl;

    }

    return 0;

}

1

1 2 3 4 5 6

4 1 2 6 5 3

2