**Question no. 1**

**#ifndef DOUBLE\_SUBSCRIPTED\_ARRAY\_H**

**#define DOUBLE\_SUBSCRIPTED\_ARRAY\_H**

**#include <iostream>**

**#include <stdexcept>**

**#include <vector>**

**class DoubleSubscriptedArray {**

**public:**

**DoubleSubscriptedArray(int rows = 1, int columns = 1)**

**: m\_rows(rows), m\_columns(columns), m\_array(rows \* columns) {}**

**// Overload () operator for non-const objects**

**int &operator()(int row, int column) {**

**return m\_array.at(index(row, column));**

**}**

**// Overload () operator for const objects**

**int operator()(int row, int column) const {**

**return m\_array.at(index(row, column));**

**}**

**// Overload == operator**

**bool operator==(const DoubleSubscriptedArray &other) const {**

**return m\_array == other.m\_array;**

**}**

**// Overload != operator**

**bool operator!=(const DoubleSubscriptedArray &other) const {**

**return !(\*this == other);**

**}**

**// Overload = operator**

**DoubleSubscriptedArray &operator=(const DoubleSubscriptedArray &other) {**

**if (this != &other) {**

**m\_rows = other.m\_rows;**

**m\_columns = other.m\_columns;**

**m\_array = other.m\_array;**

**}**

**return \*this;**

**}**

**// Overload << operator**

**friend std::ostream &operator<<(std::ostream &os,**

**const DoubleSubscriptedArray &arr) {**

**for (int i = 0; i < arr.m\_rows; ++i) {**

**for (int j = 0; j < arr.m\_columns; ++j) {**

**os << arr(i, j) << ' ';**

**}**

**os << '\n';**

**}**

**return os;**

**}**

**// Overload >> operator**

**friend std::istream &operator>>(std::istream &is,**

**DoubleSubscriptedArray &arr) {**

**for (int &val : arr.m\_array) {**

**is >> val;**

**}**

**return is;**

**}**

**private:**

**int index(int row, int column) const {**

**if (row < 0 || row >= m\_rows || column < 0 || column >= m\_columns)**

**throw std::out\_of\_range("Subscript out of range");**

**return row \* m\_columns + column;**

**}**

**int m\_rows, m\_columns;**

**std::vector<int> m\_array;**

**};**

**#endif // DOUBLE\_SUBSCRIPTED\_ARRAY\_H**

**#include "DoubleSubscriptedArray.h"**

**#include <iostream>**

**int main() {**

**DoubleSubscriptedArray chessBoard(3, 5);**

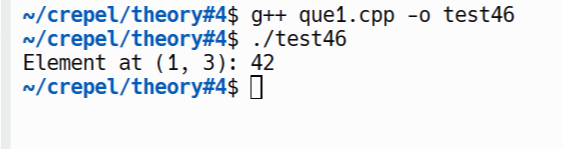
**chessBoard(1, 3) = 42;**

**std::cout << "Element at (1, 3): " << chessBoard(1, 3) << std::endl;**

**return 0;**

**}**

**Output:**

****

**Question no.2**

**#include <map>**

**#include <iostream>**

**class Polynomial {**

**private:**

**std::map<int, int> terms; // key: exponent, value: coefficient**

**public:**

**// Constructor**

**Polynomial() {}**

**// Destructor**

**~Polynomial() {}**

**// Set term**

**void setTerm(int coefficient, int exponent) {**

**terms[exponent] = coefficient;**

**}**

**// Get term**

**int getTerm(int exponent) const {**

**if (terms.find(exponent) != terms.end()) {**

**return terms.at(exponent);**

**}**

**return 0;**

**}**

**// Overload the addition operator (+)**

**Polynomial operator+(const Polynomial& p) const {**

**Polynomial result = \*this;**

**for (const auto& term : p.terms) {**

**result.terms[term.first] += term.second;**

**}**

**return result;**

**}**

**// Overload the subtraction operator (-)**

**Polynomial operator-(const Polynomial& p) const {**

**Polynomial result = \*this;**

**for (const auto& term : p.terms) {**

**result.terms[term.first] -= term.second;**

**}**

**return result;**

**}**

**// Overload the assignment operator**

**Polynomial& operator=(const Polynomial& p) {**

**if (this != &p) {**

**terms = p.terms;**

**}**

**return \*this;**

**}**

**// Overload the multiplication operator (\*)**

**Polynomial operator\*(const Polynomial& p) const {**

**Polynomial result;**

**for (const auto& term1 : terms) {**

**for (const auto& term2 : p.terms) {**

**int exponent = term1.first + term2.first;**

**int coefficient = term1.second \* term2.second;**

**result.terms[exponent] += coefficient;**

**}**

**}**

**return result;**

**}**

**// Overload the addition assignment operator (+=)**

**Polynomial& operator+=(const Polynomial& p) {**

**\*this = \*this + p;**

**return \*this;**

**}**

**// Overload the subtraction assignment operator (-=)**

**Polynomial& operator-=(const Polynomial& p) {**

**\*this = \*this - p;**

**return \*this;**

**}**

**// Overload the multiplication assignment operator (\*=)**

**Polynomial& operator\*=(const Polynomial& p) {**

**\*this = \*this \* p;**

**return \*this;**

**}**

**// Output function**

**void output() const {**

**for (auto it = terms.rbegin(); it != terms.rend(); ++it) {**

**std::cout << it->second << "x^" << it->first << " + ";**

**}**

**std::cout << "\b\b " << std::endl; // erase the last " + "**

**}**

**};**

**int main() {**

**Polynomial p1, p2;**

**// Set terms for p1**

**p1.setTerm(2, 4); // 2x^4**

**p1.setTerm(-3, -2); // -3x^-2**

**// Set terms for p2**

**p2.setTerm(3, 4); // 3x^4**

**p2.setTerm(4, -2); // 4x^-2**

**// Add p1 and p2**

**Polynomial p3 = p1 + p2;**

**// Output p3**

**p3.output();**

**// Subtract p2 from p1**

**Polynomial p4 = p1 - p2;**

**// Output p4**

**p4.output();**

**// Multiply p1 and p2**

**Polynomial p5 = p1 \* p2;**

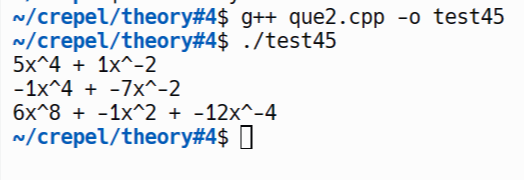
**// Output p5**

**p5.output();**

**return 0;**

**}**

**Output:**

****