**Question no.1**

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#include <ctime>

class Card {

public:

static std::string faces[13];

static std::string suits[4];

Card(int face, int suit) : face(face), suit(suit) {}

std::string toString() {

return faces[face] + " of " + suits[suit];

}

private:

int face;

int suit;

};

std::string Card::faces[] = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};

std::string Card::suits[] = {"Hearts", "Diamonds", "Clubs", "Spades"};

class DeckOfCards {

public:

DeckOfCards() {

for (int i = 0; i < 52; ++i) {

deck.push\_back(Card(i % 13, i / 13));

}

currentCard = 0;

std::srand(std::time(0));

}

void shuffle() {

currentCard = 0;

for (int i = 0; i < 52; ++i) {

int j = (std::rand() % 52);

Card temp = deck[i];

deck[i] = deck[j];

deck[j] = temp;

}

}

Card dealCard() {

if (currentCard > 51) {

shuffle();

}

if (currentCard < 52) {

return (deck[currentCard++]);

}

return (deck[0]); // Error condition, should never happen

}

bool moreCards() const {

return (currentCard < 52);

}

private:

std::vector<Card> deck;

int currentCard;

};

int main() {

DeckOfCards deckOfCards;

deckOfCards.shuffle();

while (deckOfCards.moreCards()) {

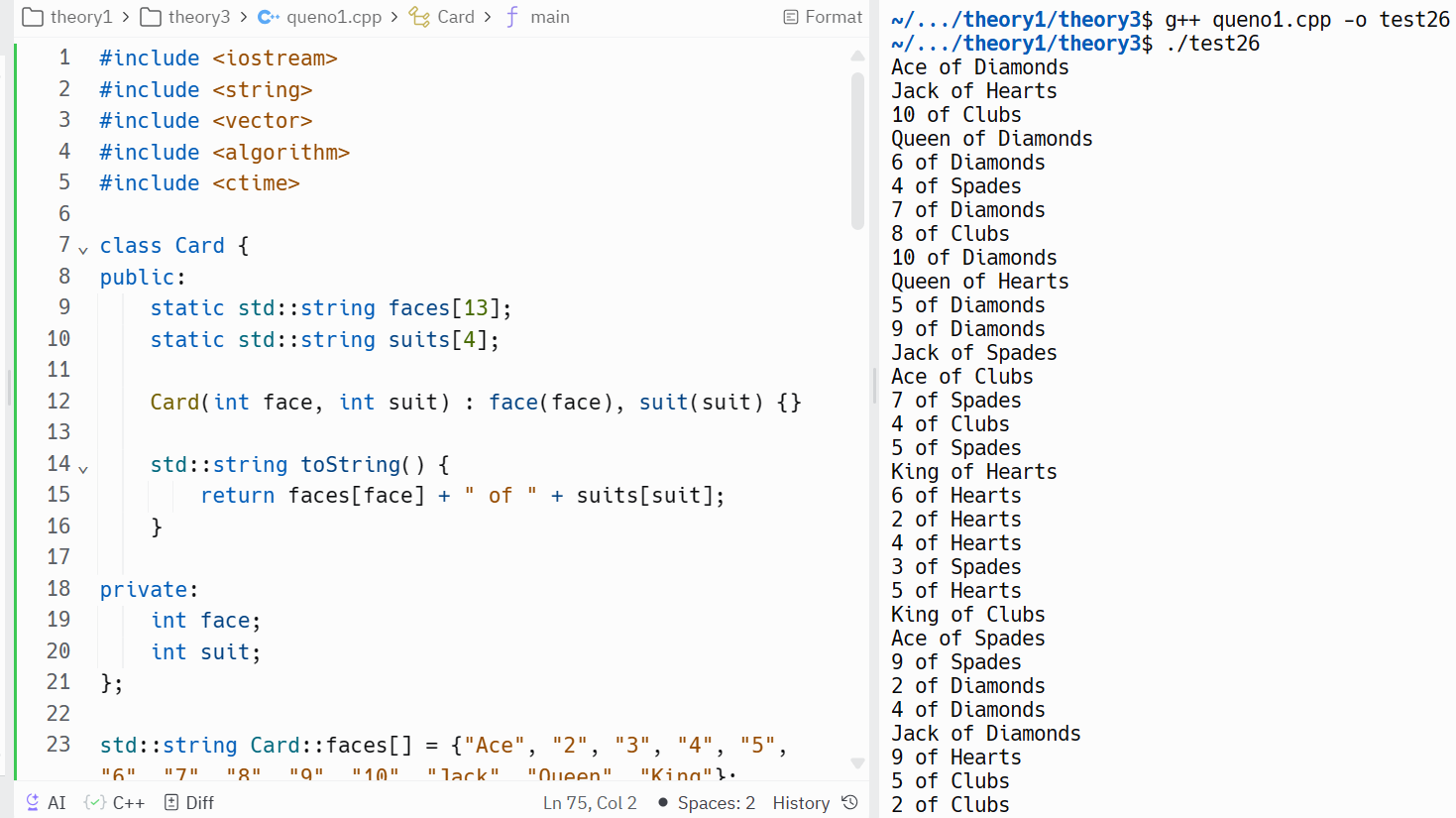
Card card = deckOfCards.dealCard();

std::cout << card.toString() << std::endl;

}

return 0;

}





Question no. 2

#include <array>

#include <iostream>

#include <vector>

using namespace std;

class IntegerSet {

private:

static const int setSize = 101;

vector<bool> setArray;

public:

IntegerSet() : setArray(setSize, false) {}

IntegerSet(const array<int, setSize> &arr) : setArray(setSize, false) {

for (int i : arr) {

if (i >= 0 && i < setSize)

setArray[i] = true;

}

}

IntegerSet unionOfSets(const IntegerSet &set2) const {

IntegerSet unionSet;

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

unionSet.setArray[i] = setArray[i] || set2.setArray[i];

}

return unionSet;

}

IntegerSet intersectionOfSets(const IntegerSet &set2) const {

IntegerSet intersectionSet;

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

intersectionSet.setArray[i] = setArray[i] && set2.setArray[i];

}

return intersectionSet;

}

void insertElement(int k) {

if (k >= 0 && k < setSize)

setArray[k] = true;

}

void deleteElement(int m) {

if (m >= 0 && m < setSize)

setArray[m] = false;

}

void printSet() const {

bool isEmpty = true;

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

if (setArray[i]) {

cout << i << " ";

isEmpty = false;

}

}

if (isEmpty)

cout << "---";

cout << endl;

}

bool isEqualTo(const IntegerSet &set2) const {

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

if (setArray[i] != set2.setArray[i])

return false;

}

return true;

}

};

int main() {

// Testing IntegerSet class

IntegerSet set1;

IntegerSet set2({1, 3, 5, 7, 9});

IntegerSet set3({2, 4, 6, 8, 10});

IntegerSet set4({1, 3, 5, 7, 9});

cout << "Set 1: ";

set1.printSet();

cout << "Set 2: ";

set2.printSet();

cout << "Set 3: ";

set3.printSet();

cout << "Set 4: ";

set4.printSet();

cout << "Union of Set 2 and Set 3: ";

(set2.unionOfSets(set3)).printSet();

cout << "Intersection of Set 2 and Set 3: ";

(set2.intersectionOfSets(set3)).printSet();

cout << "Is Set 2 equal to Set 4? " << (set2.isEqualTo(set4) ? "Yes" : "No")

<< endl;

cout << "Is Set 2 equal to Set 3? " << (set2.isEqualTo(set3) ? "Yes" : "No")

<< endl;

set1.insertElement(2);

set1.insertElement(4);

set1.insertElement(6);

set1.insertElement(8);

set1.insertElement(10);

cout << "Set 1 after insertion: ";

set1.printSet();

set1.deleteElement(2);

set1.deleteElement(6);

cout << "Set 1 after deletion: ";

set1.printSet();

return 0;

}

