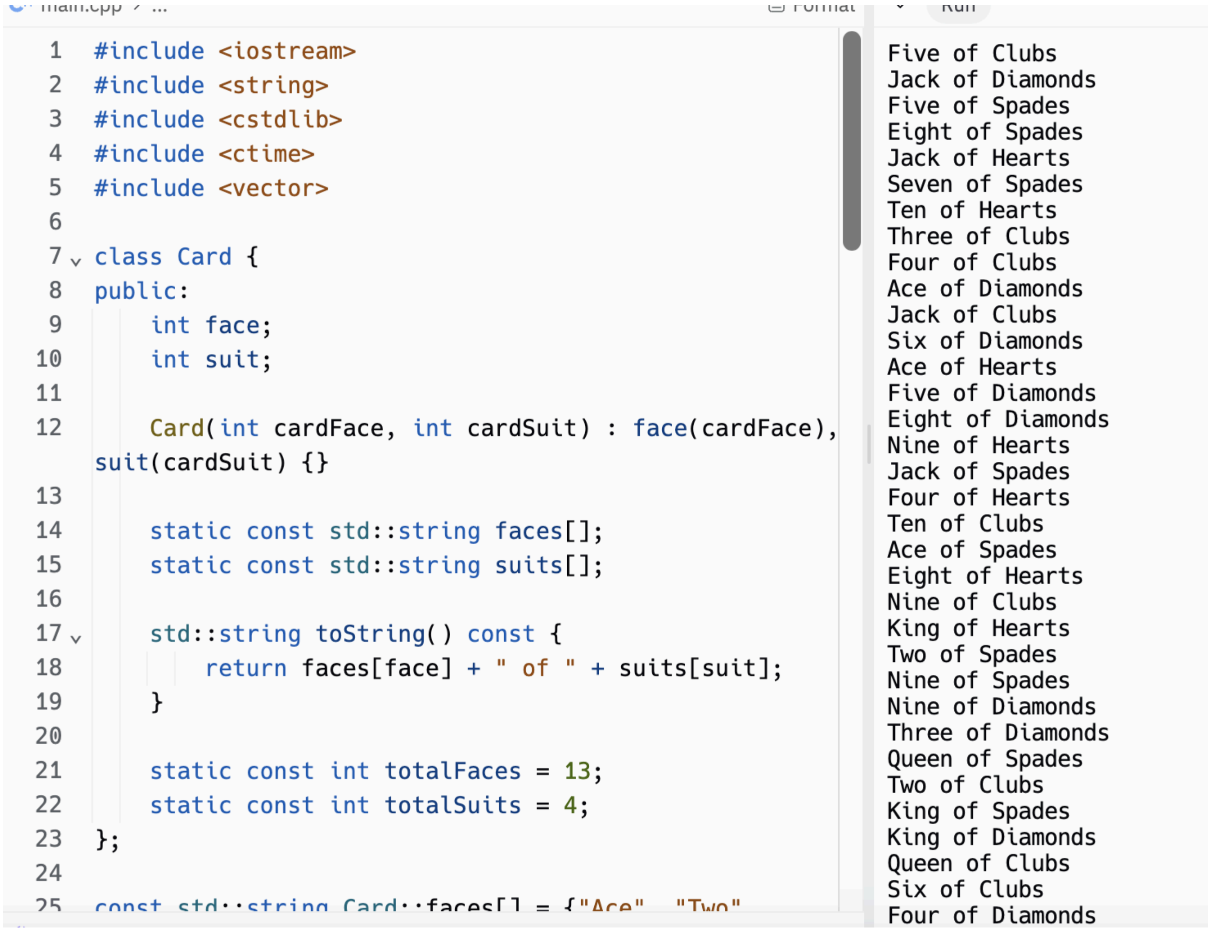
Q1)

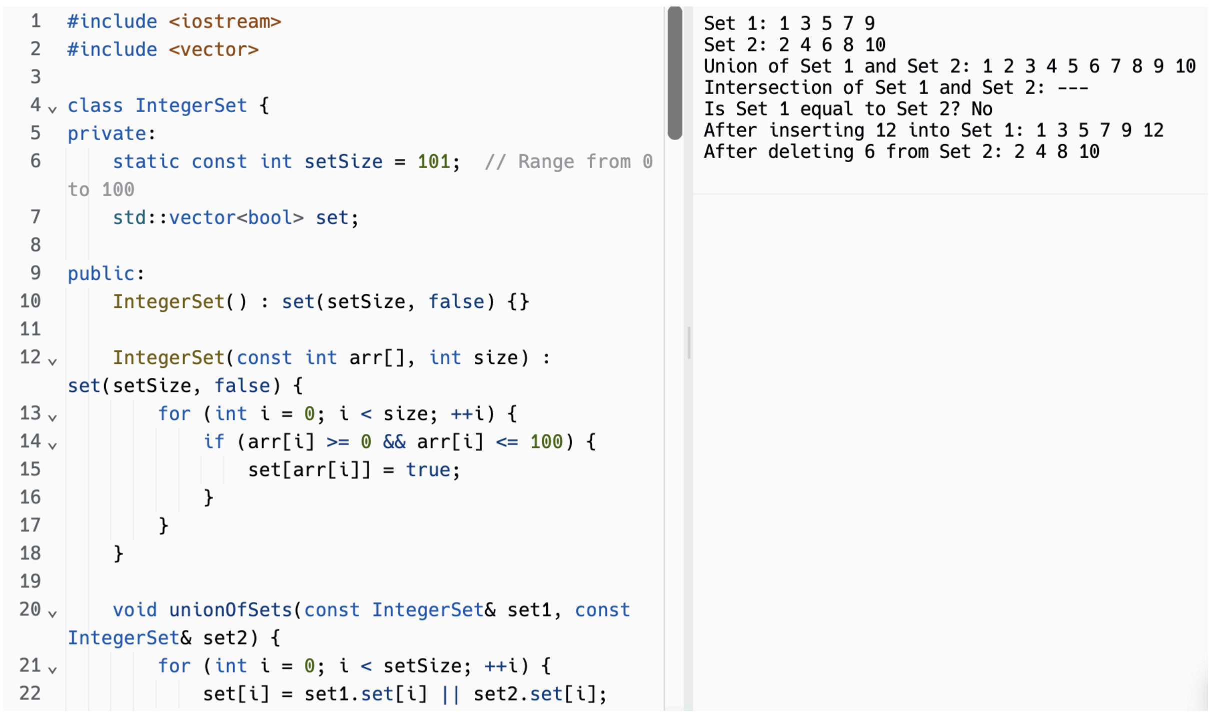


```cpp
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <vector>

class Card {
public:
    int face;
    int suit;

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

    static const std::string faces[];
    static const std::string suits[];

    std::string toString() const {
        return faces[face] + " of " + suits[suit];
    }

    static const int totalFaces = 13;
    static const int totalSuits = 4;
};

const std::string Card::faces[] = {"Ace", "Two"
```

```
Five of Clubs
Jack of Diamonds
Five of Spades
Eight of Spades
Jack of Hearts
Seven of Spades
Ten of Hearts
Three of Clubs
Four of Clubs
Ace of Diamonds
Jack of Clubs
Six of Diamonds
Ace of Hearts
Five of Diamonds
Eight of Diamonds
Nine of Hearts
Jack of Spades
Four of Hearts
Ten of Clubs
Ace of Spades
Eight of Hearts
Nine of Clubs
King of Hearts
Two of Spades
Nine of Spades
Nine of Diamonds
Three of Diamonds
Queen of Spades
Two of Clubs
King of Spades
King of Diamonds
Queen of Clubs
Six of Clubs
Four of Diamonds
```

```cpp
//Q.no.1

#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <vector>

class Card {
public:
    int face;
    int suit;

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

    static const std::string faces[];
    static const std::string suits[];

    std::string toString() const {
        return faces[face] + " of " + suits[suit];
    }

    static const int totalFaces = 13;
    static const int totalSuits = 4;
};

const std::string Card::faces[] = {"Ace", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight",
"Nine", "Ten", "Jack", "Queen", "King"};
const std::string Card::suits[] = {"Hearts", "Diamonds", "Clubs", "Spades"};

class DeckOfCards {
public:
    static const int totalCards = 52;
    std::vector<Card> deck;
    int currentCard;

    DeckOfCards() : currentCard(0) {
        for (int i = 0; i < Card::totalSuits; ++i) {
            for (int j = 0; j < Card::totalFaces; ++j) {
                deck.push_back(Card(j, i));
            }
        }
    }

    void shuffle() {
        for (int i = 0; i < totalCards; ++i) {
            int randIndex = rand() % totalCards;
            std::swap(deck[i], deck[randIndex]);
        }
    }

    Card dealCard() {
        return deck[currentCard++];
    }

    bool moreCards() const {
        return currentCard < totalCards;
    }
};

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));

    DeckOfCards myDeck;
    myDeck.shuffle();

    while (myDeck.moreCards()) {
        Card dealtCard = myDeck.dealCard();
        std::cout << dealtCard.toString() << std::endl;
    }

    return 0;
}
```

Q2)



```
#include <iostream>
#include <vector>

class IntegerSet {
private:
    static const int setSize = 101;  // Range from 0 to 100
    std::vector<bool> set;

public:
    IntegerSet() : set(setSize, false) {}

    IntegerSet(const int arr[], int size) : set(setSize, false) {
        for (int i = 0; i < size; ++i) {
            if (arr[i] >= 0 && arr[i] <= 100) {
                set[arr[i]] = true;
            }
        }
    }

    void unionOfSets(const IntegerSet& set1, const IntegerSet& set2) {
        for (int i = 0; i < setSize; ++i) {
            set[i] = set1.set[i] || set2.set[i];
```

```
Set 1: 1 3 5 7 9
Set 2: 2 4 6 8 10
Union of Set 1 and Set 2: 1 2 3 4 5 6 7 8 9 10
Intersection of Set 1 and Set 2: ---
Is Set 1 equal to Set 2? No
After inserting 12 into Set 1: 1 3 5 7 9 12
After deleting 6 from Set 2: 2 4 8 10
```