/\*To create ADT that implement the "set" concept. a. Add (newElement) -Place a value into the set b. Remove (element) Remove the value c. Contains (element) Return true if element is in collection d. Size () Return number of values in collection Iterator () Return an iterator used to loop over collection e. Intersection of two sets f. Union of two sets g. Difference between two sets h.Subset\*/

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

class Set {

private:

vector<T> elements;

public:

// Add element to set

void add(T element) {

if (!contains(element))

elements.push\_back(element);

}

// Remove element from set

void remove(T element) {

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

if (\*it == element) {

elements.erase(it);

return;

}

}

}

// Check if element exists

bool contains(T element) const {

for (const auto& el : elements) {

if (el == element)

return true;

}

return false;

}

// Return number of elements

int size() const {

return elements.size();

}

// Iterator

typename vector<T>::iterator begin() { return elements.begin(); }

typename vector<T>::iterator end() { return elements.end(); }

// Print elements (helper)

void print() const {

cout << "{ ";

for (const auto& el : elements) {

cout << el << " ";

}

cout << "}" << endl;

}

// Intersection

Set<T> intersection(const Set<T>& other) const {

Set<T> result;

for (const auto& el : elements) {

if (other.contains(el))

result.add(el);

}

return result;

}

// Union

Set<T> unionSet(const Set<T>& other) const {

Set<T> result;

for (const auto& el : elements)

result.add(el);

for (const auto& el : other.elements)

result.add(el);

return result;

}

// Difference

Set<T> difference(const Set<T>& other) const {

Set<T> result;

for (const auto& el : elements) {

if (!other.contains(el))

result.add(el);

}

return result;

}

// Subset

bool isSubset(const Set<T>& other) const {

for (const auto& el : elements) {

if (!other.contains(el))

return false;

}

return true;

}

};

int main() {

Set<int> setA, setB;

setA.add(1);

setA.add(2);

setA.add(3);

setB.add(3);

setB.add(4);

setB.add(5);

cout << "Set A: "; setA.print();

cout << "Set B: "; setB.print();

Set<int> inter = setA.intersection(setB);

Set<int> uni = setA.unionSet(setB);

Set<int> diff = setA.difference(setB);

cout << "\nIntersection: "; inter.print();

cout << "Union: "; uni.print();

cout << "Difference (A - B): "; diff.print();

cout << "\nIs A a subset of B? " << (setA.isSubset(setB) ? "Yes" : "No") << endl;

// Iterating using iterator

cout << "Iterating Set A: ";

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

cout << \*it << " ";

}

cout << endl;

return 0;

}

OUTPUT

Set A: { 1 2 3 }

Set B: { 3 4 5 }

Intersection: { 3 }

Union: { 1 2 3 4 5 }

Difference (A - B): { 1 2 }

Is A a subset of B? No

Iterating Set A: 1 2 3