/*Implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.
1 2
0.3 4
0.5 6
0.2 8
Enter 2nd set
0.5 2
0.4 4
0.1 6
1 8
*/

```
#include<bits/stdc++.h>
#include <chrono>
using namespace std;

class FuzzySet{
	public:
	vector<pair<double,int>> arr1;
	vector<pair<double,int>> arr2;
	vector<pair<double,int>> union_arr;
	vector<pair<double,int>> inter_arr;
	vector<pair<double,int>> arr1_comp;
	vector<pair<double,int>> arr2_comp;
	vector<pair<double,int>> diff1;
	vector<pair<double,int>> diff2;

	int n;
	FuzzySet(){
		cout<<"Enter number if elements\n";cin>>n;
		cout<<"Enter 1st set\n";
		for(int i=0;i<n;i++)
		{
			double x,y;
			cin>>x>>y;
			arr1.push_back(make_pair(x,y));
		}
		cout<<"Enter 2nd set\n";
		for(int i=0;i<n;i++)
		{
			double x,y;
			cin>>x>>y;
			arr2.push_back(make_pair(x,y));
		}

	}
	//FUZZY relation R = A X B
	void fuzz_relation(){
	vector<vector<double>> fuzz_rel(n,vector<double>(n,0));
		for(int i=0;i<n;i++)
```

```
	{
		for(int j=0;j<n;j++)
		{
			if(arr1[i].first<=arr2[j].first)
			{
				fuzz_rel[i][j]=arr1[i].first;
			}
			else{
				fuzz_rel[i][j]=arr2[j].first;
			}
		}
	}

	cout<<"FUZZY RELATION IS : \n";
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			cout<<fuzz_rel[i][j]<<" ";
		}
		cout<<"\n";
	}
}

void display(){
	for(auto x:arr1)
	{
		cout<<x.first<<" ";
	}
	cout<<"\n";
	for(auto x:arr1)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
	for(auto x:arr1)
	{
		cout<<x.second<<"    ";
	}
	cout<<"\n\n";

	for(auto x:arr2)
	{
		cout<<x.first<<" ";
	}
	cout<<"\n";
	for(auto x:arr2)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
	for(auto x:arr2)
```

```
		{
			cout<<x.second<<"    ";
		}
		cout<<"\n";
	}

	//Union ->  degree_of_membership(Y)= max(degree_of_membership(A),
degree_of_membership(B))
	void union_fuzz(){
		for(int i=0;i<n;i++)
		{
			if(arr1[i].first>=arr2[i].first)
			{
				union_arr.push_back(arr1[i]);
			}
			else{
				union_arr.push_back(arr2[i]);
			}
		}
		cout<<"Union is : \n";
		for(auto x:union_arr)
		{
			cout<<x.first<<"    ";
		}
		cout<<"\n";
		for(auto x:union_arr)
		{
			cout<<"----"<<" ";
		}
		cout<<"\n";
		for(auto x:union_arr)
		{
			cout<<x.second<<"    ";
		}
		cout<<"\n";
	}

	void inter_fuzz(){
		for(int i=0;i<n;i++)
		{
			if(arr1[i].first>=arr2[i].first)
			{
				inter_arr.push_back(arr2[i]);
			}
			else{
				inter_arr.push_back(arr1[i]);
			}
		}
		cout<<"Intersection is : \n";
		for(auto x:inter_arr)
		{
			cout<<x.first<<"    ";
```

```
	}
	cout<<"\n";
	for(auto x:inter_arr)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
	for(auto x:inter_arr)
	{
		cout<<x.second<<"    ";
	}
	cout<<"\n";
}

void complement_fuzz(){
	for(int i=0;i<n;i++)
	{
		double ans=1-arr1[i].first;
		arr1_comp.push_back(make_pair(ans,arr1[i].second));
	}
	for(int i=0;i<n;i++)
	{
		double ans=1-arr2[i].first;
		arr2_comp.push_back(make_pair(ans,arr1[i].second));
	}
	cout<<"Complement of arr1 :\n";
	for(auto x:arr1_comp)
	{
		cout<<x.first<<"    ";
	}
	cout<<"\n";
	for(auto x:arr1_comp)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
	for(auto x:arr1_comp)
	{
		cout<<x.second<<"    ";
	}
	cout<<"\n";
	cout<<"Complement of arr2 :\n";
	for(auto x:arr2_comp)
	{
		cout<<x.first<<"    ";
	}
	cout<<"\n";
	for(auto x:arr2_comp)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
```

```
	for(auto x:arr2_comp)
	{
		cout<<x.second<<"    ";
	}
	cout<<"\n";

}


void diff_fuzz(){

//diff is min(A,complement of B);

	for(int i=0;i<n;i++)
	{
		if(arr1[i].first>=arr2_comp[i].first)
		{
			diff1.push_back(arr2_comp[i]);
		}
		else{
			diff1.push_back(arr1[i]);
		}
	}
	cout<<"Difference is : \n";
	for(auto x:diff1)
	{
		cout<<x.first<<"    ";
	}
	cout<<"\n";
	for(auto x:diff1)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
	for(auto x:diff1)
	{
		cout<<x.second<<"    ";
	}
	cout<<"\n";

	//Diff B|A = min(B,comple(A))

	for(int i=0;i<n;i++)
	{
		if(arr2[i].first>=arr1_comp[i].first)
		{
			diff2.push_back(arr1_comp[i]);
		}
		else{
			diff2.push_back(arr2[i]);
		}
	}
```

```
	cout<<"Difference is : \n";
	for(auto x:diff2)
	{
		cout<<x.first<<"    ";
	}
	cout<<"\n";
	for(auto x:diff2)
	{
		cout<<"----"<<" ";
	}
	cout<<"\n";
	for(auto x:diff2)
	{
		cout<<x.second<<"    ";
	}
	cout<<"\n";
}

void max_min_compo(){
int m1,n1,m2,n2;

cout<<"Enter size of 1st\n";cin>>m1>>n1;
vector<vector<double>> fun1(m1,vector<double>(n1,0));

cout<<"Enter Fuzzy relation 1\n";
for(int i=0;i<m1;i++)
{
	for(int j=0;j<n1;j++)
	{
		cin>>fun1[i][j];
	}
}
cout<<"Enter size of 2nd\n";cin>>m2>>n2;
vector<vector<double>> fun2(m2,vector<double>(n2,0));
cout<<"Enter Fuzzy relation 2\n";
for(int i=0;i<m2;i++)
{
	for(int j=0;j<n2;j++)
	{
		cin>>fun2[i][j];
	}
}

vector<vector<double>> res(m1,vector<double>(n2,0));
for(int i=0;i<m1;i++)
{
	for(int j=0;j<n2;j++)
	{
		for(int k=0;k<n1;k++)
		{
			res[i][j] = max(res[i][j],min(fun1[i][k],fun2[k][j]));
		}
```

```
                }
        }
        cout<<"MAX MIN COMPOSITION IS :\n";
        for(int i=0;i<m1;i++)
        {
                for(int j=0;j<n2;j++)
                {
                        cout<<res[i][j]<<" ";
                }
                cout<<"\n";
        }


        }


};

int main(){
        FuzzySet f;
        auto t_start = std::chrono::high_resolution_clock::now();
        // the work...
        auto t_end = std::chrono::high_resolution_clock::now();
        f.union_fuzz();
        f.inter_fuzz();
        f.complement_fuzz();
        f.diff_fuzz();
        f.fuzz_relation();
        f.max_min_compo();
        double elapsed_time_ms = std::chrono::duration<double, std::milli>(t_end-t_start).count();
        cout<<"time : "<<elapsed_time_ms<<endl;
}

//output



(base) omkar@omkar-lenovo:~/OMKAR/BE/LP4/a1$ g++ a1.cpp
(base) omkar@omkar-lenovo:~/OMKAR/BE/LP4/a1$ ./a.out
Enter number if elements
4
Enter 1st set
1 2
0.3 4
0.5 6
0.2 8
Enter 2nd set
0.5 2
0.4 4
0.1 6
1 8
Union is :
```

1 0.4 0.5 1
---- ---- ---- ----
2 4 6 8
Intersection is :
0.5 0.3 0.1 0.2
---- ---- ---- ----
2 4 6 8
Complement of arr1 :
0 0.7 0.5 0.8
---- ---- ---- ----
2 4 6 8
Complement of arr2 :
0.5 0.6 0.9 0
---- ---- ---- ----
2 4 6 8
Difference is :
0.5 0.3 0.5 0
---- ---- ---- ----
2 4 6 8
Difference is :
0 0.4 0.1 0.8
---- ---- ---- ----
2 4 6 8
FUZZY RELATION IS :
0.5 0.4 0.1 1
0.3 0.3 0.1 0.3
0.5 0.4 0.1 0.5
0.2 0.2 0.1 0.2
Enter size of 1st
1 3
Enter Fuzzy relation 1
0.1 0.2 0.7
Enter size of 2nd
3 3
Enter Fuzzy relation 2
0.9 0.4 0.9
0.2 0.2 0.2
0.5 0.4 0.5
MAX MIN COMPOSITION IS :
0.5 0.4 0.5
time : 0.000508