ÖDEV 4:

CEVAPLAR

1. Kesirli sayılar, *a / b* formundadır. Burada, *a* ve *b* tam sayılardır(integers) ve $b \neq 0$. a / b ve c / d kesirli sayılar olsun. Kesirli sayılarla ilgili aşağıdaki kurallar tanımlıdır:

$$a/b + c/d = (ad + bc)/bd$$
$$a/b - c/d = (ad - bc)/bd$$
$$a/b \times c/d = ac/bd$$
$$(a/b)/(c/d) = ad/bc, in\ which\ c/d \neq 0$$

Kesirli sayılar, şöyle karşılaştırılır: *a / b* op *c / d* eğer *ad* op *bc*, burada op ilgili karşılaştırma operatörünü belirtmektedir. Örneğin, *a / b* < *c / d* eğer *ad* < *bc*.

Fraction adında bir sınıf (class) tasarlayın. Bu sınıf, kesirli sayılarla ilgili aritmetik ve karşılaştırma işlemleri gerçekleştirebilecektir. Sınıf, +, -, *, / şeklinde aritmetik operatörler ve <, >, ==, != şeklinde karşılaştırma operator fonksiyonlarına sahip olacaktır. Ayrıca, the stream insertion << ve stream extraction >> operatörleri de girdi ve çıktı işlemleri için kullanılabilecektir.

class Fraction için sınıf kodlarını yazınız.

Tüm operatörleri test etmek üzere, Fraction sınıfını kullanan bir program yazınız. Bu programda, tüm operator testleri yanında aşağıdaki testi de gerçekleştirin: *x*, *y*, ve *z* Fraction nesneleri olsun. Eğer, *2/3* klavyeden girdi olarak verilirse, :

```
cin >> x;
```

komutu *2/3* değerini *x* nesnesine yükleyecektir. Aşağıdaki komut:

```
cout << x + y << endl;
```

x + y işlem sonucunu kesirli formda ekrana yazdıracaktır. Aşağıdaki komut:

```
z = x + y;
```

x ve *y* toplamını *z* nesnesine kesirli formda atayacaktır.

Cevap:

```
/// Fraction.h
#include <iostream>
using namespace std;

class Fraction
{
public:
	friend ostream& operator<<(ostream& out, const Fraction&
fraction);
	friend istream& operator>>(istream& in, Fraction& fraction);

	Fraction(int numerator = 0 , int denumerator = 1);

	Fraction operator+(const Fraction& fraction);
	Fraction operator-(const Fraction& fraction);
	Fraction operator*(const Fraction& fraction);
	Fraction operator/(const Fraction& fraction);

	bool operator<(const Fraction& fraction);
	bool operator>(const Fraction& fraction);
	bool operator==(const Fraction& fraction);
	bool operator!=(const Fraction& fraction);

	void PerformReduction();
private:
	int num;
	int den;


	int GCD(int number1,int number2);
};

/// Fraction.cpp
#include "Fraction.h"



Fraction::Fraction(int numerator, int
denumerator):num(numerator),den(denumerator)
{
}

Fraction Fraction::operator+(const Fraction& fraction)
{
	Fraction newfraction(num * fraction.den + den * fraction.num,
                                                den * fraction.den);

	newfraction.PerformReduction();
```

```
	return newfraction;

}

Fraction Fraction::operator-(const Fraction& fraction)
{
	Fraction newfraction(num * fraction.den - den * fraction.num,
	                                          den * fraction.den);

	newfraction.PerformReduction();

	return newfraction;
}

Fraction Fraction::operator*(const Fraction& fraction)
{
	Fraction newfraction(num * fraction.num , den * fraction.den);

	newfraction.PerformReduction();

	return newfraction;
}

Fraction Fraction::operator/(const Fraction& fraction)
{
	Fraction newfraction(num * fraction.den, den * fraction.num);

	newfraction.PerformReduction();

	return newfraction;
}

bool Fraction::operator<(const Fraction& fraction)
{
	if ((num * fraction.den) < (den * fraction.num))
		return true;

	return false;
}

bool Fraction::operator>(const Fraction& fraction)
{
	if ((num * fraction.den) > (den * fraction.num))
		return true;

	return false;
}

bool Fraction::operator==(const Fraction& fraction)
{
	if ((num * fraction.den) == (den * fraction.num))
		return true;
```

```
	return false;
}

bool Fraction::operator!=(const Fraction& fraction)
{
	return !(*this == fraction);
}

void Fraction::PerformReduction()
{
	int divisor = GCD(num,den);

	num /= divisor;
	den /= divisor;
}

int Fraction::GCD(int number1, int number2)
{
	int lowerBount;
	int greatestCommonDivisor = 1;

	if (number1 < number2) lowerBount = number1;
	else lowerBount = number2;

	for (int divisor = 2; divisor <= lowerBount; divisor++)
	{
		if( (number1 % divisor == 0) && (number2 % divisor == 0) )
		{
			greatestCommonDivisor = divisor;
		}
	}
	return greatestCommonDivisor;
}

ostream& operator<<(ostream& out, const Fraction& fraction)
{
	out << "(" << fraction.num << "/" << fraction.den << ")";
	return out;
}

istream& operator>>(istream& in, Fraction& fraction)
{
	in >> fraction.num >> fraction.den;
	return in;
}

/// FractionTest.cpp
#include "Fraction.h"

int main()
{
```

```
Fraction x,y,z;

cout << "INPUT TEST" << endl;
cout << "Enter numerator & denumerator for Fraction 'x' ) : ";
cin >> x;
cout << "Enter numerator & denumerator for Fraction 'y' ) : ";
cin >> y;

cout << "OUTPUT TEST" << endl;
cout << " 'x' = " << x << endl;
cout << " 'y' = " << y << endl;

cout << "ADDITION TEST" << endl;
z = x + y;
cout << " 'z = x + y' = " << z << endl;

cout << "SUBTRACTION TEST" << endl;
z = x - y;
cout << " 'z = x - y' = " << z << endl;

cout << "MULTIPLICATION TEST" << endl;
z = x * y;
cout << " 'z = x * y' = " << z << endl;

cout << "DIVISON TEST" << endl;
z = x / y;
cout << " 'z = x / y' = " << z << endl;

Fraction f1(1, 2);
Fraction f2(2, 4);
Fraction f3(2, 3);

cout << "EQUALITY TEST" << endl;
if (f1 == f2)
      cout << f1 << "==" << f2 << endl;
else
      cout << f1 << "!=" << f2 << endl;

if (f1 == f3)
      cout << f1 << "==" << f3 << endl;
else
      cout << f1 << "!=" << f3 << endl;

cout << "INEQUALITY TEST" << endl;
if (f1 != f2)
      cout << f1 << "!=" << f2 << endl;
else
      cout << f1 << "==" << f2 << endl;

if (f1 != f3)
      cout << f1 << "!=" << f3 << endl;
else
```

```
        cout << f1 << "==" << f3 << endl;

    cout << "GREATER TEST" << endl;
    if (f1 > f3)
        cout << f1 << ">" << f3 << endl;
    else
        cout << f1 << "not >" << f3 << endl;

    if (f3 > f1)
        cout << f3 << ">" << f1 << endl;
    else
        cout << f3 << "not >" << f1 << endl;

    cout << "SMALLER TEST" << endl;

    if (f1 < f3)
        cout << f1 << "<" << f3 << endl;
    else
        cout << f1 << "not <" << f3 << endl;

    if (f3 < f1)
        cout << f3 << "<" << f1 << endl;
    else
        cout << f3 << "not <" << f1 << endl;

}
```

2. Buz pateni turnuvasında 10 hakem, puanlarını 0.0-6.0 arasında verir. Bir patencinin ortalama puanı hakemlerin verdiği en yüksek ve en düşük puanların toplam puandan çıkartılıp, hesaplanmasıyla bulunur. Kış olimpiyatlarında kullanılmak üzere aşağıdaki bileşenlerden oluşan bir program yazınız.

- Sporcu sınıfı:
  - Veri üyeleri: isim ve ulke (private olacak)
  - Üye fonksiyonlar: veri üyelerini atayan ve döndüren fonksiyonlar. Sporcunun ismini ve ülkesini ekrana yazdıran print fonksiyonu. Üyelerin değerlerini parametre olarak alan yapıcı (constructor) fonksiyon.
- Sporcu sınıfından türetilmiş Patenci sınıfı:
  - Veri üyeleri: skor dizisi ve ortalama puan (private olacak)
  - Üye fonksiyonlar: veri üyelerini atayan ve döndüren fonksiyonlar. Patencinin ismini, ülkesini, skorlarını ve ortalama puanını ekrana yazdıran print fonksiyonu. Üyelerin değerlerini (isim ve ülke değerlerini de) parametre olarak alan yapıcı (constructor) fonksiyon.

a) Sporcu sınıfını "Sporcu.h" dosyasında kodlayınız.
b) Sporcu sınıfını "Sporcu.cpp" dosyasında kodlayınız.
c) Patenci sınıfını "Patenci.h" dosyasında kodlayınız.

d) Patenci sınıfını "Patenci.cpp" dosyasında kodlayınız.
e) main() fonksiyonunu "sporApp.cpp" dosyasında aşağıdaki çıktıyı üretecek şekilde kodlayınız.

Örnek Çıktı:

```
Patencinin ismini giriniz: Mehmet ↲
Patencinin ülkesini giriniz: Turkey ↲
Skorları giriniz: 5.5 5.6 4.3 5.9 4.6 5.5 5.7 5.4 4.9 5.7 ↲


Isim: Mehmet
Ulke: Turkey
Skorları: 5.5 5.6 4.3 5.9 4.6 5.5 5.7 5.4 4.9 5.7
Ortalama: 5.36
```

Cevap:

(a)

```
/// Sporcu.h
#include <iostream>
#include <string>
using namespace std;

class Sporcu{
public:
	Sporcu(const string& isim, const string& ulke);


	string IsimAl() const;
	void IsimAta(const string& isim);

	string UlkeAl() const;
	void UlkeAta(const string& ulke);

	void print() const;
private:
	string isim;
	string ulke;

};
```

(b)

```
/// Sporcu.cpp

#include "Sporcu.h"



Sporcu::Sporcu(const string& _isim, const string& _ulke) :
isim(_isim),ulke(_ulke)
{
}

string Sporcu::IsimAl() const
{
	return isim;
}

void Sporcu::IsimAta(const string& isim)
{
	isim = isim;
}

string Sporcu::UlkeAl() const
{
	return ulke;
}

void Sporcu::UlkeAta(const string& ulke)
{
	ulke = ulke;
}

void Sporcu::print() const
{
	cout << "Isim : " << isim << endl
		<< "Ulke : " << ulke << endl;
}
```

(c)

```
/// Patenci.h
#include "Sporcu.h"
class Patenci :public Sporcu{
public:
	Patenci(const string& isim, const string& ulke, float skor[10]);


	float skorAl(int index) const;
	void skorAta(int index, float skor);

	float ortalamaPuanAl() const;
	void ortalamaPuanAta(float ortalamaPuan);
```

```
	void print() const;

private:
	float skor[10];
	float ortalamaPuan;
};
```

(d)

```
/// Patenci.cpp
#include "Patenci.h"
#include <iomanip>

Patenci::Patenci(const string& _isim, const string& _ulke, float
_skor[10]) : Sporcu(_isim,_ulke)
{
	for (int i = 0; i < 10; i++)
	{
		skor[i] = _skor[i];
	}

	float min = skor[0];
	float max = skor[0];
	float sum = 0.0;

	for (int i = 0; i < 10; i++){
		if (skor[i] < min)
			min = skor[i];
		if (skor[i] > max)
			max = skor[i];
		sum += skor[i];
	}

	sum -= min;
	sum -= max;

	ortalamaPuan = sum / 8;

}

float Patenci::skorAl(int index) const
{
	if (index < 0 || index > 10)
		return -1;

	return skor[index];
}

void Patenci::skorAta(int index, float skor)
{
```

```
	if (index < 0 || index > 10)
		return;

	skor[index] = skor;
}

float Patenci::ortalamaPuanAl() const
{
	return ortalamaPuan;
}

void Patenci::ortalamaPuanAta(float ortalamaPuan)
{
	this->ortalamaPuan = ortalamaPuan;
}

void Patenci::print() const
{
	Sporcu::print();

	cout << "Skorlari : ";
	cout << setprecision(2);

	for (int i = 0; i < 10; i++){
		cout << skor[i] << " ";
	}
	cout << endl;
	cout << "Ortalama : " << setprecision(3) << ortalamaPuan << endl;

}
```

(e)

```
 /// sporcuApp.cpp
#include "Patenci.h"

int main(){
	string isim, ulke;
	float skorlar[10];

	cout << "Patencinin ismini giriniz: ";
	cin >> isim;
	cout << "Patencinin ulkesini giriniz: ";
	cin >> ulke;

	cout << "Skorlari giriniz: ";
	for (int i = 0; i < 10; i++){
		cin >> skorlar[i];
	}
	Patenci p(isim, ulke, skorlar);
	p.print();
 }
```