Lab04 04/16

Please complete the following classes and test them with the given main program.

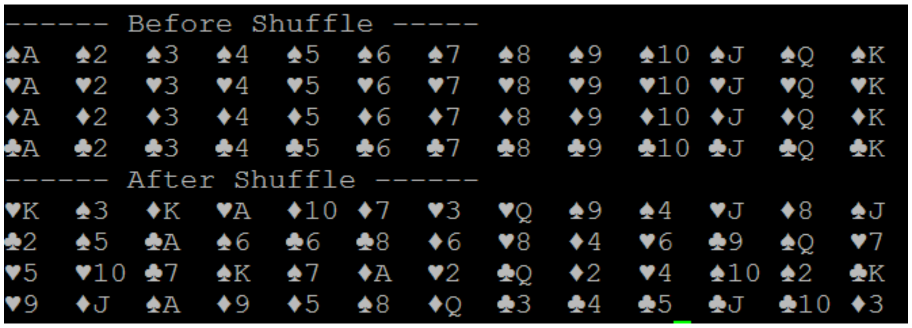
**** Do **NOT** modify the given programs. ****

1. A Deck of Cards

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

char SUIT[4][13] =
{"\xE2\x99\xA0","\xE2\x99\xA5","\xE2\x99\xA6","\xE2\x99\xA3"};
char NUM[13][3] = {"A","2","3","4","5","6","7","8","9","10","J","Q","K"};

class Card {
        private:
                char suit[13];
                short num;

        public:
                Card() { /* nothing */ };

                // Create a Card with given suit and number
                Card(char* su, short nu);

                // Set a blank card's suit and number
                void set(char* su, short nu);

                // Swap a Card itself with another Card (tar)
                void swap(Card& tar);

                // To print a Card on screen
                void show();
};
```

```
class A_Deck_Of_Cards {
        private:
                Card* cards;
        public:
                // Initialize “cards” with dynamic array of 52 Cards
                // with their own suits and numbers
                A_Deck_Of_Cards();

                // shuffle “cards”
                void shuffle();

                // Display the cards on the screen
                void show();
};

/* Main Function */
int main() {
        srand(time(NULL));

        A_Deck_Of_Cards Dcards;

        puts("------ Before Shuffle -----");
        Dcards.show();
        Dcards.shuffle();
        puts("------ After Shuffle ------");
        Dcards.show();

        return 0;
}
```

Sample Output:



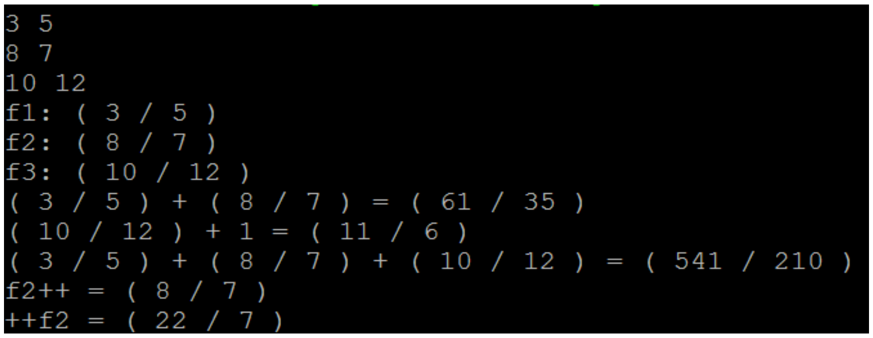
```
------ Before Shuffle -----
♠A  ♠2  ♠3  ♠4  ♠5  ♠6  ♠7  ♠8  ♠9  ♠10 ♠J  ♠Q  ♠K
♥A  ♥2  ♥3  ♥4  ♥5  ♥6  ♥7  ♥8  ♥9  ♥10 ♥J  ♥Q  ♥K
♦A  ♦2  ♦3  ♦4  ♦5  ♦6  ♦7  ♦8  ♦9  ♦10 ♦J  ♦Q  ♦K
♣A  ♣2  ♣3  ♣4  ♣5  ♣6  ♣7  ♣8  ♣9  ♣10 ♣J  ♣Q  ♣K
------ After Shuffle ------
♥K  ♠3  ♦K  ♥A  ♦10 ♦7  ♥3  ♥Q  ♠9  ♠4  ♥J  ♦8  ♠J
♣2  ♠5  ♣A  ♠6  ♣6  ♣8  ♦6  ♥8  ♦4  ♥6  ♣9  ♠Q  ♥7
♥5  ♥10 ♣7  ♠K  ♠7  ♦A  ♥2  ♣Q  ♦2  ♥4  ♠10 ♠2  ♣K
♥9  ♦J  ♠A  ♦9  ♦5  ♠8  ♦Q  ♣3  ♣4  ♣5  ♣J  ♣10 ♦3
```

2. Fraction

```
#include <iostream>
using namespace std;

class Fraction {
        private:
                int numer;
                int denom;

                // Calculate the "greatest common divisor" of the two integers
                int gcd(int, int);

                // Calculate the "least common multiple" of the two integers
                int lcm(int, int);

        public:
                // If the initial values are not assigned, just set it as 0
                Fraction() { numer = 0; denom = 1; };

                // Set the fraction's numerator and denominator
                void set(int n, int d);

                // Reduce the fraction to the lowest terms
                // Hint: You may use the gcd and lcm
                void reduce();

                // Overload operator+
                // Implement the addtion of two fractions
                // aka. Fraction + Fraction
                Fraction operator+(const Fraction &);

                // Overload operator+
                // Implement the addtion of a fractions and an integer
                // aka. Fraction + integer
                Fraction operator+(int);
```

```
		// Overload ++f
		Fraction operator++();

		// Overload f++
		Fraction operator++(int);

		// Overload input stream of a fraction
		friend istream& operator>>(istream&, Fraction&);

		// Overload output stream of a fraction
		friend ostream& operator<<(ostream&, const Fraction&);
};

int main() {
	Fraction f1,f2,f3;

	cin>>f1>>f2>>f3;
	cout<<"f1: "<<f1<<endl;
	cout<<"f2: "<<f2<<endl;
	cout<<"f3: "<<f3<<endl;

	cout<<f1<<" + "<<f2<<" = "<<(f1 + f2)<<endl;
	cout<<f3<<" + "<<"1"<<" = "<<(f3 + 1)<<endl;
	cout<<f1<<" + "<<f2<<" + "<<f3<<" = "<<(f1 + f2 + f3)<<endl;
	cout<<"f2++ = "<<(f2++)<<endl;
	cout<<"++f2 = "<<(++f2)<<endl;

	return 0;
}
```

Sample Output:

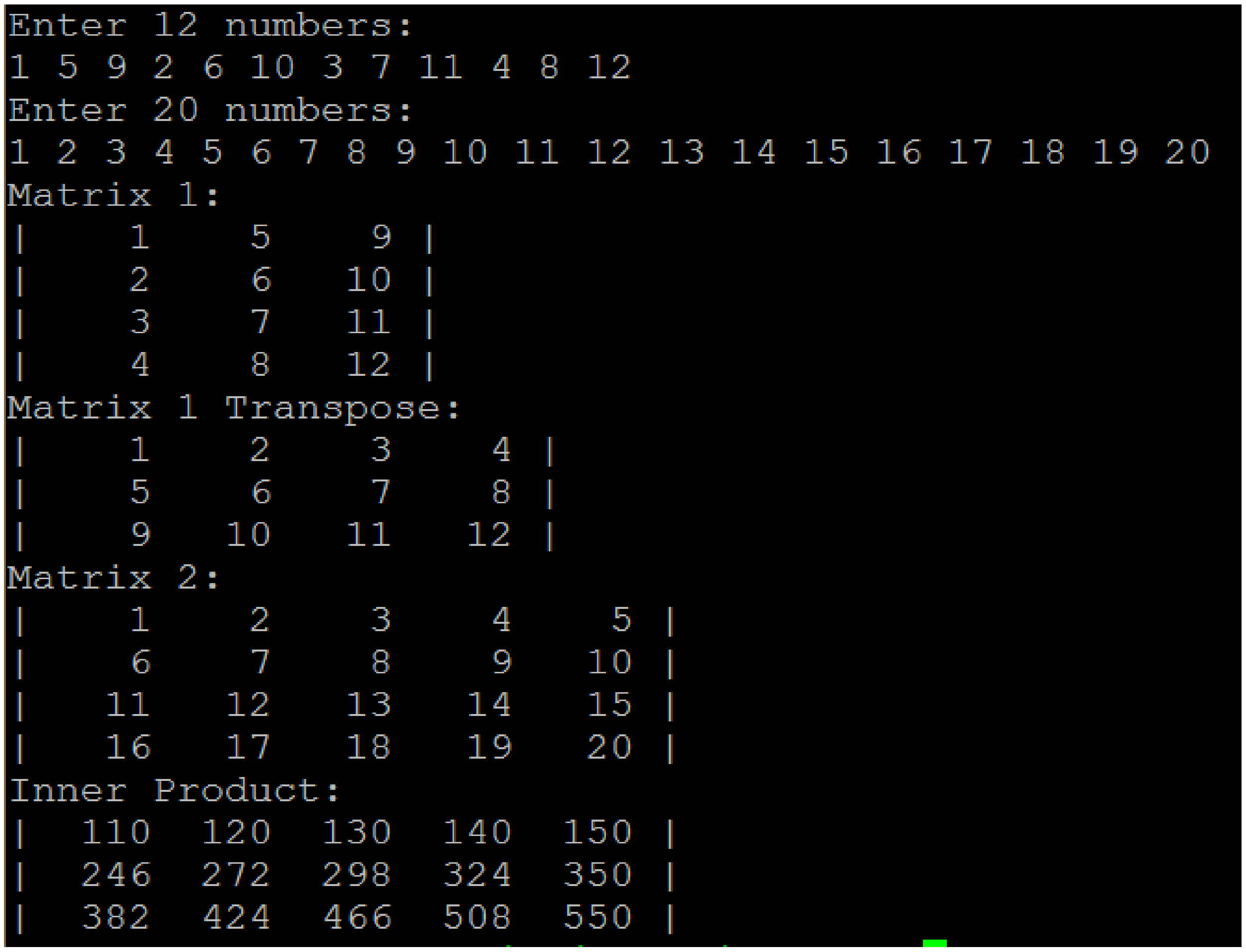


```
3 5
8 7
10 12
f1: ( 3 / 5 )
f2: ( 8 / 7 )
f3: ( 10 / 12 )
( 3 / 5 ) + ( 8 / 7 ) = ( 61 / 35 )
( 10 / 12 ) + 1 = ( 11 / 6 )
( 3 / 5 ) + ( 8 / 7 ) + ( 10 / 12 ) = ( 541 / 210 )
f2++ = ( 8 / 7 )
++f2 = ( 22 / 7 )
```

3. Matrix

```
#include <stdio.h>

class Matrix {
	private:
		int row,col;
		int** content;

	public:
		// Create a n x m Matrix as a dynamic array
		// Initalize the Matrix as a zero matrix
		Matrix(int n,int m);

		// Give a value to each entry of the matrix with a series of numbers
		void input(int M[]);

		// Display the matrix on the screen
		void display();

		// Return its transposed matrix
		Matrix transpose();

		// Return the product of two Matrices
		Matrix operator*(const Matrix&);
};

int main(){

	// Sample Data
	int M1[] = {1,5,9,2,6,10,3,7,11,4,8,12};
	int M2[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};

	Matrix m1(4,3), m2(4,5);
	m1.input(M1);
	m2.input(M2);

	puts("Matrix 1:");
	m1.display();
	puts("Matrix 1 Transpose:");
	m1.transpose().display();
	puts("Matrix 2:");
	m2.display();
	puts("Inner Product:");
	(m1.transpose()*m2).display();

	return 0;
}
```

Sample Output:



```
Enter 12 numbers:
1 5 9 2 6 10 3 7 11 4 8 12
Enter 20 numbers:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Matrix 1:
|    1    5    9 |
|    2    6   10 |
|    3    7   11 |
|    4    8   12 |
Matrix 1 Transpose:
|    1    2    3    4 |
|    5    6    7    8 |
|    9   10   11   12 |
Matrix 2:
|    1    2    3    4    5 |
|    6    7    8    9   10 |
|   11   12   13   14   15 |
|   16   17   18   19   20 |
Inner Product:
|  110  120  130  140  150 |
|  246  272  298  324  350 |
|  382  424  466  508  550 |
```