```c
#include <stdio.h>
#include <stdlib.h>
#include "racionais.h"
#define SEED 0
/* Arquivo MAIN que usa o TAD racionais */

/* coloque seus includes aqui */
int ler(int max)
{
    int numero;
    do
    {
        scanf("%d", &numero);

    } while (numero > max && numero > 0);

    return numero;
}

int main()
{
    srand(SEED);
    int n, max;
    struct racional r1, r2;

    n = ler(100);
    max = ler(30);

    for (int i = 1; i <= n; i++)
    {
        printf("%d: ", i);

        r1 = sorteia_r(max);
        r2 = sorteia_r(max);

        imprime_r(r1);
        imprime_r(r2);

        if (!valido_r(r1) || !valido_r(r2))
        {
            printf("NUMERO INVALIDO\n");
            return 1;
        }

        /*pela logica das saidas do professor
        nao eh impresso soma, sub, mul, div
        se o resultado da
        divisao for invalida*/
        struct racional divisao = divide_r(r1, r2);
        if (!valido_r(divisao))
        {
            printf("DIVISAO INVALIDA\n");
            return 1;
        }

        imprime_r(soma_r(r1, r2));
        imprime_r(subtrai_r(r1, r2));
        imprime_r(multiplica_r(r1, r2));
        imprime_r(divisao);

        printf("\n");
    }

    return 0;
}
```

linha de espaço
idem

este comentário deveria ter sido trocado por um explicando a função

então posso digitar um negativo???

mistura entre comando (srand) e declarações de variáveis

os espaços deveriam ser impressos aqui

```c
#include <stdio.h>
#include <stdlib.h>
#include "racionais.h"

int aleat(int min, int max)
{
    if (max < min)
    {
        /*garantindo que max seja maior*/
        int troca = max;
        max = min;
        min = troca;
    }

    return min + (rand() % (max - min + 1));
}

int mdc(int a, int b)
{
    if (b == 0)
        return a;
    return mdc(b, a % b);
}

int mmc(int a, int b)
{
    return ((a * b) / mdc(a, b));
}

struct racional simplifica_r(struct racional r)
{
    if (valido_r(r))
    {
        int divisor_comum;
        divisor_comum = mdc(r.num, r.den);
        r.num /= divisor_comum;
        r.den /= divisor_comum;

        /*perceba que se o denominador estiver negativo,
        vou ter que multiplicar o numerador por -1,
        nÃ£o importando o sinal do numerador*/
        if (r.den < 0)
        {
            r.num *= -1;
            r.den *= -1;
        }
    }
    return r;
}
struct racional cria_r(int numerador, int denominador)
{
    struct racional r;

    r.num = numerador;
    r.den = denominador;
    r.valido = valido_r(r);

    return r;
}

struct racional sorteia_r(int n)
{
    struct racional r;
    int denominador, numerador;

    numerador = aleat(0, n);
    denominador = aleat(0, n);

    r = cria_r(numerador, denominador);

    return r;
}
void imprime_r(struct racional r)
{
```

ambos

```
    r = simplifica_r(r);
    if (!valido_r(r))
    {
        printf("INVALIDO ");
        return;
    }
    else if (r.den == r.num)
    {
        printf("1 ");
        return;
    }
    else if (!r.num)
    {
        printf("0 ");
        return;
    }
    else if (r.den == 1)
    {
        printf("%d ", r.num);
        return;
    }
    else
        printf("%d/%d ", r.num, r.den);
}
int valido_r(struct racional r)
{
    if (!r.den)
        return 0;
    return 1;
}

struct racional soma_r(struct racional r1, struct racional r2)
{
    if (valido_r(r1) && valido_r((r2)))
    {
        if (r1.den == r2.den)
        {
            /*se os denominadores forem iguais, Ã© sÃ³ fazer a soma
            dos numeradores*/
            r1.num += r2.num;
        }
        else
        {
            /*caso contrario, calcula o mmc dos denominadores,
            divide pelo denominador de r1 vezes o numerador de r1
            mais o resultado de
            mmc dividido pelo numerador de r2 vezes o numerador de r2*/
            int mmc_var = mmc(r1.den, r2.den);
            r1.num = ((mmc_var / r1.den) * r1.num) + ((mmc_var / r2.den) * r2.num);
            r1.den = mmc_var;
        }

        return r1;
    }
    /*aterrando, garantindo que o retorno seja invalido*/
    r1.valido = 0;
    r1.den = 0;
    return r1;
}
struct racional subtrai_r(struct racional r1, struct racional r2)
{
    if (valido_r(r1) && valido_r((r2)))
    {
        if (r1.den == r2.den)
        {
            r1.num -= r2.num;
            return r1;
        }
        else
        {
            /*exceto o sinal, mesma coisa que a soma*/
            int mmc_var = mmc(r1.den, r2.den);
            r1.num = ((mmc_var / r1.den) * r1.num) - ((mmc_var / r2.den) * r2.num);
```

com os return não precisa dos else.

a especificação diz para não ter espaços

```
            r1.den = mmc_var;
        }

        return r1;
    }
    r1.valido = 0;
    r1.den = 0;
    return r1;
}
struct racional multiplica_r(struct racional r1, struct racional r2)
{
    if (valido_r(r1) && valido_r((r2)))
    {
        r1.num *= r2.num;
        r1.den *= r2.den;

        r1.valido = valido_r(r1);

        return r1;
    }

    r1.valido = 0;
    r1.den = 0;
    return r1;
}
struct racional divide_r(struct racional r1, struct racional r2)
{

    if (valido_r(r1) && valido_r((r2)))
    {
        int mutz;
        mutz = r2.den;            aux, não mutz!!!!
        r2.den = r2.num;
        r2.num = mutz;

        r1 = multiplica_r(r1, r2);

        if (!r1.den)
        {
            r1.valido = 0;
            r1.den = 0;
            return r1;
        }

        r1.valido = 1;
        return r1;
    }
    r1.valido = 0;
    r1.den = 0;
    return r1;
}
```