```c
/*
 * Tipos Abstratos de Dados - TADs
 * Arquivo de implementação para TAD racional.
 * Feito em 16/09/2024 para a disciplina CI1001 - Programação 1.
 *
 * Este arquivo deve conter as implementações das funções cujos protótipos
 * foram definidos em racional.h. Neste arquivo também podem ser definidas
 * funções auxiliares para facilitar a implementação daquelas funções.
*/

/* coloque aqui seus includes (primeiro os <...>, depois os "...") */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>      para que?
#include "racional.h"


/*
 * Implemente aqui as funcoes definidas no racionais.h; caso precise,
 * pode definir aqui funcoes auxiliares adicionais, que devem ser usadas
 * somente neste arquivo.
*/

/* retorna um número aleatório entre min e max, inclusive. */
long aleat(long min, long max) {

  /* Gera um número aleatório entre min e max, inclusive */
  return min + rand() % (max - min + 1);
}

/* Máximo Divisor Comum entre a e b      */
/* calcula o MDC pelo método de Euclides */
long mdc (long a, long b){
  a = a;
  b = b;
  /* Valores sempre positivos, visto que há possibilidade de entrarem negativos*/

  if (!a)
      return b;
  else if (!b)
      return a;

  if(a > b)
      return mdc(a%b,b);
  return mdc(a,b%a);
}

/* Mínimo Múltiplo Comum entre a e b */
/* mmc = (a * b) / mdc (a, b)        */
long mmc (long a, long b){

  return (a * b) / mdc (a, b);
}

/* Recebe um número racional e o simplifica.
 * Por exemplo, ao receber 10/8 deve retornar 5/4.
 * Se ambos numerador e denominador forem negativos, deve retornar um positivo.
 * Se o denominador for negativo, o sinal deve migrar para o numerador.
 * Se r for inválido, devolve-o sem simplificar. */
struct racional simplifica_r (struct racional r){
  struct racional aux;
  long aux_mdc;

  aux_mdc = mdc(labs(r.num),labs(r.den));
  aux.num = r.num / aux_mdc;
  aux.den = r.den / aux_mdc;

  return aux;
}

/* implemente as demais funções aqui */

/* função de teste*/    ???
struct racional cria_r (long numerador, long denominador){
  struct racional aux;

  aux.num = numerador;
  aux.den = denominador;
  aux = simplifica_r(aux);

  return aux;
}
```

```c
struct racional sorteia_r (long min, long max){
  struct racional aux;

  aux.num = aleat(min,max);
  aux.den = aleat(min,max);
  aux = simplifica_r(aux);

  return aux;
}

void imprime_r (struct racional r){

  if(!valido_r(r)){
    printf("INVALIDO ");
    return;
  }

  if (r.num == 0)
    printf("0 ");

  else if(r.den == 1 || r.den == -1)
    printf("%ld ", r.num/r.den);

  else if(r.num == r.den)
    printf("1 ");

  else{
    if(r.num > 0 && r.den < 0)
        printf("-%ld/%ld ", labs(r.num), labs(r.den));
    else if(r.num < 0 && r.den < 0)
        printf("%ld/%ld ", labs(r.num), labs(r.den));
    else
        printf("%ld/%ld ", 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){
  struct racional aux;
  long aux_mmc;

  aux_mmc = mmc(labs(r1.den),labs(r2.den));
  aux.num = ((aux_mmc / r1.den * r1.num) + (aux_mmc / r2.den * r2.num));
  aux.den = aux_mmc;
  aux = simplifica_r(aux);

  return aux;
}

struct racional subtrai_r (struct racional r1, struct racional r2){
  struct racional aux;
  long aux_mmc;

  aux_mmc = mmc(labs(r1.den),labs(r2.den));
  aux.num = ((aux_mmc / r1.den * r1.num) - (aux_mmc / r2.den * r2.num));
  aux.den = aux_mmc;
  aux = simplifica_r(aux);

  return aux;
}

struct racional multiplica_r (struct racional r1, struct racional r2){
  struct racional aux;

  aux.num = r1.num * r2.num;
  aux.den = r1.den * r2.den;
  aux = simplifica_r(aux);
  return aux;
}

struct racional divide_r (struct racional r1, struct racional r2){
  struct racional aux;

  aux.num = r1.num * r2.den;
  aux.den = r1.den * r2.num;
```

estava especificado para não imprimir espaços nesta função!

```
  aux = simplifica_r(aux);
  return(aux);
}
```

```c
/*
 * Tipos Abstratos de Dados - TADs
 * Arquivo do programa principal, que usa o TAD racional.
 * Feito em 16/09/2024 para a disciplina CI1001 - Programação 1.
*/

/* coloque aqui seus includes (primeiro os <...>, depois os "...") */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "racional.h"

/* programa principal */
int main (){

    srand (0); /* use assim, com zero */
    int n, max;
    struct racional r1, r2, soma, sub, mult, div;

    do{
        printf("Digite um numero entre 0 e 100:");
        scanf("%d",&n);
        if (n < 0 || n > 100)
            printf("Numero n fora do intervalo definido\n");
    }while(n < 0 || n > 100);

    do{
        printf("Digite outro numero entre 0 e 30:");
        scanf("%d",&max);
        if(max < 0 || max > 30)
            printf("Maximo fora do intervalo definido\n");
    }while(max < 0 || max > 30);
    /* Garantia de que os numeros lidos estão no intervalos pré determinados*/


    for(int i = 1; i <= n; i++){
        printf("%d: ",i);
        r1 = sorteia_r(-max,max);
        r2 = sorteia_r(-max,max);

        imprime_r(r1);
        imprime_r(r2);

        if(!valido_r(r1) || !valido_r(r2)){

            printf("NUMERO INVALIDO\n");
            return 1;
            /* Retorno de erro para o programa*/
        }

        soma = soma_r(r1,r2);
        sub = subtrai_r(r1,r2);
        mult = multiplica_r(r1,r2);
        div = divide_r(r1,r2);

        if(!valido_r(div)){
            printf("DIVISAO INVALIDA\n");
            return 1;
            /* Retorno de erro para o programa*/
        }

        imprime_r(soma);
        imprime_r(sub);
        imprime_r(mult);
        imprime_r(div);
        printf("\n");
    }

    return 0;
}
```

declarações de variáveis antes de comandos.