Exame de Programação I

(Época de recurso, duração: 2 horas)

Universidade do Algarve

5 de Fevereiro de 2003

(2 valores) Pergunta 1. Faça uma função para calcular o máximo divisor comum (mdc) entre dois números inteiros não negativos. Para tal, utilize o algoritmo de Euclides que é definido do seguinte modo:

$$mdc\ (\ a,\ b\ ) = \begin{cases} a & \text{se } b = 0 \\ mdc\ (\ b,\ a\%b\ ) & \text{se } b > 0 \end{cases}$$

em que o sinal % tem o significado usual em C (resto da divisão inteira). Exemplo de aplicação do algoritmo de Euclides para calcular o mdc entre 30 e 21.

```
mdc(30,21) = mdc(21,9)
           = mdc(9,3)
           = mdc(3,0)
           = 3
```

(6 valores) Pergunta 2. Imagine que para além dos tipos básicos (`char`, `int`, `float`, `double`), a linguagem C tinha também o tipo `racional` para guardar números racionais (isto é, fracções tais como `2/3` ou `12/5`). Imagine também que a linguagem C tinha uma biblioteca chamada `racional.h` contendo as seguintes 3 funções:

```
/* constrói um número racional a partir de um numerador e
   de um denominador */
racional constroi_racional( int num, int den );

/* devolve o numerador de um número racional */
int numer( racional r );

/* devolve o denominador de um número racional */
int denom( racional r );
```

Estas 3 funções servem de base para podermos fazer outras funções que permitam manipular números racionais. Por exemplo, podemos fazer uma função que permita escrever um número racional do seguinte modo:

```
void escreve_racional( racional r )
{
  printf("%d/%d", numer(r), denom(r) );
}
```

Escreva funções semelhantes para calcular a soma, diferença, multiplicação, e divisão de números racionais. Para lhe facilitar a vida, junto se anexa o cabeçalho das funções. Assim, basta preencher o "miolo".

```
/* adiciona r1 com r2 */
racional soma_racional( racional r1, racional r2 )
{
  ...
}

/* subtrai r2 de r1 */
racional subtrai_racional( racional r1, racional r2 )
{
  ...
}

/* multiplica r1 por r2 */
racional mult_racional( racional r1, racional r2 )
{
  ...
}

/* divide r1 por r2 */
racional div_racional( racional r1, racional r2 )
{
  ...
}
```

(2 valores) Pergunta 3 Altere a função `escreve_racional` de modo a que o número seja escrito no ecrã de forma simplifica. Por exemplo, se o numerador for 2 e o denominador for 6, aquilo que deve ser escrito no ecrã é `1/3`. (DICA: Pode (e deve) utilizar a função `mdc` feita na pergunta 1.)

(1+3=4 valores) Pergunta 4 Nas perguntas anteriores, assumimos a existência de uma biblioteca chamada `racional.h`, mas infelizmente essa biblioteca não existe na linguagem C. Contudo, isso não é grave porque podemos definir e implementar as coisas que a biblioteca `racional.h` estava a fazer por nós. Assim,

(a) defina um tipo de dados que permita representar um número `racional`.

(b) implemente as funções `constroi_racional`, `numer`, e `denom`.

(3 valores) Pergunta 5 Faça um programa que receba como input um numero inteiro n e que escreva como output, todos os pares de números inteiros positivos (i, j) tais que $i + j$ seja um número primo e $1 \leq i < j \leq n$. Por exemplo, se $n = 6$, o output seria:

```
(1,2) --> 3
(1,4) --> 5
(1,6) --> 7
(2,3) --> 5
(2,5) --> 7
(3,4) --> 7
(5,6) --> 11
```

NOTA: Para este exercício, pode assumir que existe uma função chamada `primo`, nos mesmos moldes que foi feita nas aulas teórico-práticas.

(2+1=3 valores) Pergunta 6. Considere a função `sqr` e a função `misterio` definidas do seguinte modo.

```
/* calcula o quadrado de x */
int sqr( int x ) { return x*x; }

/* ... */
int misterio( int x, int n )
{
  if( n==0 )
    return 1;
  else if( n%2==0 )
    return sqr( misterio(x, n/2) );
  else
    return x * misterio(x, n-1);
}
```

(a) Preencha o comentário para a função `misterio` (não mais do que 10 palavras).

(b) Qual o output de `misterio(3,5)`?