Computação e Programação

(LEAmb, LEBl, LEGI, LEGM, LEMat, LEQ, LQ)

Departamento de Matemática, IST

Exame 1

13 de Janeiro de 2006, 9h00

Duração: 2h30

não preencher
I –
II –
III – ___
T:

Curso: _____ Número: _____ Nome: _____

Grupo I [1,5+1,5]

a) Defina em Matlab uma função `f` que quando recebe como argumento um vector de números inteiros percorre esse vector posição a posição recorrendo a um ciclo `while` e devolve a soma dos elementos do vector que ocorrem imediatamente antes de um número par.

b) Defina em Matlab uma função `g` que quando recebe como argumento uma matriz de números inteiros percorre essa matriz posição a posição recorrendo a dois ciclos `while` encaixados e devolve um vector com tantas posições quantas as linhas da matriz contendo na posição k a soma dos elementos da linha k da matriz dada que ocorrem imediatamente acima dum número par.

Grupo II [2,0+2,0]

O jogo do galo joga-se num tabuleiro de 3×3. Dois jogadores jogam alternadamente escolhendo uma casa livre e marcando-a com **X** (primeiro jogador) ou **O** (segundo jogador). O primeiro jogador que conseguir ocupar as três casas duma linha, coluna ou diagonal vence o jogo.

a) Desenvolva em F um módulo que disponibilize as operações a seguir descritas sobre o tabuleiro do jogo do galo, escolhendo a implementação seguinte para o tipo `jogo`.

```
type, public :: jogo
  private
  character(len=1), dimension(3,3) :: tab
  character(len=1) :: seg
end type jogo
```

Com esta implementação, pretende-se que o tabuleiro seja representado por uma matriz 3×3 em que as casas estão numeradas da seguinte forma.

$(1,1)$	$(1,2)$	$(1,3)$
$(2,1)$	$(2,2)$	$(2,3)$
$(3,1)$	$(3,2)$	$(3,3)$

Um elemento do tipo `jogo` contém em cada posição da matriz a letra do jogador que ocupou a casa correspondente (**X** ou **O**), ou a string vazia "" se a casa ainda estiver livre. O campo `seg` indica a letra do próximo jogador a jogar.

As operações disponibilizadas pelo módulo são as seguintes:

- `novo()`: função que devolve um resultado de tipo `jogo` em que todas as casas estão livres e o próximo jogador a jogar é **X**;
- `jogada(g,i,j)`: subrotina que recebe no parâmetro de entrada/saída `g` um tabuleiro e nos parâmetros de entrada `i` e `j` dois números inteiros entre 1 e 3 e devolve em `g` o resultado de o próximo jogador jogar na casa (`i`,`j`) caso esta esteja livre, alterando o jogador que deve jogar a seguir;
- `seg(g)`: função que recebe um argumento `g` de tipo `jogo` e devolve uma string de comprimento 1 contendo a letra do próximo jogador a jogar;
- `livre(g,i,j)`: função que recebe um jogo `g` e dois inteiros `i` e `j` entre 1 e 3 e devolve o valor lógico `.true.` se a casa (`i`,`j`) estiver livre e `.false.` caso contrário;
- `val(g,i,j)`: função que recebe um jogo `g` e dois inteiros `i` e `j` entre 1 e 3 e devolve a letra do jogador que ocupou a casa (`i`,`j`) caso esta esteja ocupada;
- `ganha(g,c)`: função que recebe no argumento `g` um jogo e no argumento `c` a letra de um dos jogadores e devolve `.true.` se o tabuleiro corresponder a uma configuração em que o jogador com a letra `c` venceu o jogo.

b) Desenvolva em F, recorrendo ao módulo acima, uma função `legal(g)` que recebe um jogo `g` e devolve `.true.` ou `.false.` consoante este `g` representa ou não uma posição que pode ser obtida através de uma sequência válida de jogadas. Note que isto corresponde a verificar as duas condições seguintes:

- ou ambos os jogadores ocupam o mesmo número de casas e o próximo a jogar é **X** ou **X** tem mais uma casa ocupada que **O** e o próximo a jogar é **O**;
- se um dos jogadores ganhou, é a vez do outro jogar.

Grupo III [2,0+1,0]

a) Considere a seguinte função `g`, definida em F. Descreva a funcionalidade desta função. Defina recursivamente uma função `h` com a mesma funcionalidade de `g`.

```
function g(v) result(y)
integer, dimension(:), intent(in) :: v
integer :: y
integer :: i,j

i=1
j=2
y=0
do
   if (i>size(v)) then
      exit
   else if (j>size(v)) then
      y=y+v(i)
      exit
```

```
    else
       y=y+v(i)+v(j)
       i=i+2
       j=j+2
    end if
 end do
 end function g
```

b) Seja f uma função contínua num intervalo $[a, b]$ satisfazendo $f(a)f(b) < 0$. Então f tem uma raíz no intervalo $[a, b]$, isto é, existe um valor x com $a < x < b$ tal que $f(x) = 0$.

Um dos métodos utilizados para encontrar valores aproximados de uma dessas raízes é o método da bissecção, que a seguir se descreve.

Em primeiro lugar definem-se sucessões $\{a_n\}$ e $\{b_n\}$ da seguinte forma:

1. $a_0 = a$ e $b_0 = b$;
2. dados a_n e b_n:
   (a) toma-se $m = (a_n + b_n)/2$;
   (b) se $f(a_n)f(m) < 0$, então $a_{n+1} = a_n$ e $b_{n+1} = m$;
   (c) caso contrário, $a_{n+1} = m$ e $b_{n+1} = b_n$;

O método consiste em fixar um valor e para o erro pretendido para a aproximação e calcular valores sucessivos de a_n e b_n até que $b_n - a_n < e$. Quando esta condição se verifica, a_n (ou b_n) é a aproximação pretendida para a raíz.

Implemente o método da bissecção em F através de uma função `bissec` que receba como argumentos os reais `a`, `b` (extremos do intervalo) e `e` (erro máximo da aproximação) e a função real de variável real `f` e devolva a aproximação encontrada para a raíz da equação `f(x)`=0.