FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Licenciatura em Engenharia Informática e Computação

Introdução à Programação de Computadores I

Exame 1ª Chamada, 12 de Janeiro de 2001

DURAÇÃO MÁXIMA 2 horas e 30 minutos, com consulta

Aluno | Nº

Problema 1 **(7.5 valores)**

Vejamos uma forma de definir a *função de Ackermann*, em que *m* e *n* são inteiros não negativos:

$$A(m,\ n) = \begin{cases} 0 & se\ \ n = 0 \\ 2 * n & se\ \ m = 0 \\ 2 & se\ \ n = 1 \\ A(m-1,\ A(m,\ n-1)) & para\ \ os\ \ outros\ \ casos \end{cases}$$

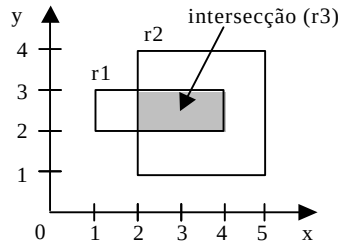
1.1 Escrever em *Scheme* o procedimento *ackermann* que devolve o valor da *função de Ackermann*, sabendo que os argumentos serão sempre inteiros não negativos.

1.2 Escrever o procedimento *ackermann-geral* que devolve o valor da função de *Ackermann* se ambos os argumentos forem inteiros não negativos, e devolve -1 se algum dos argumentos for inteiro negativo. O procedimento *ackermann-geral* deve usar o procedimento *ackermann*.

Problema 2 (7.5 valores)

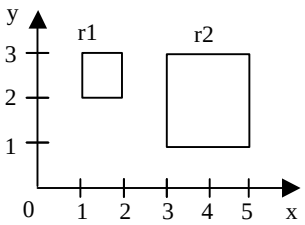
Pretende-se escrever parte de um programa em *Scheme* que determina a intersecção de dois rectângulos, suportando diálogos do seguinte tipo (os dados introduzidos pelo utilizador estão em **negrito**):

```
Coordenadas do rectangulo 1 (xmin xmax ymin ymax):
1 4 2 3
Coordenadas do rectângulo 2 (xmin xmax ymin ymax):
2 5 1 4
Coordenadas do rectangulo resultante:
2 4 2 3
```



Vejamos outro diálogo:

```
Coordenadas do rectangulo 1 (xmin xmax ymin ymax):
1 2 2 3
Coordenadas do rectangulo 2 (xmin xmax ymin ymax):
3 5 1 3
Os rectangulos nao se intersectam.
```



Um rectângulo será criado com o construtor *faz-rectangulo*.

```
(define faz-rectangulo
  (lambda (xmin xmax ymin ymax)
    (list xmin xmax ymin ymax)))
```

2.1- Escrever o procedimento ***interseccao-rect***, que calcula e devolve o rectângulo (r3) resultante da intersecção de dois rectângulos passados como argumento (r1 e r2). Se os rectângulos não se intersectam, devolve a lista vazia '**()**. Não é da responsabilidade deste procedimento dialogar com o utilizador (isso estaria a cargo do procedimento principal do programa, que não é feito aqui).

Pista: A intersecção de dois rectângulos pode ser determinada com base no cálculo de máximos e mínimos das coordenadas dos rectângulos... *xmin* será o máximo entre os *xmin* dos 2 rectângulos... *xmax* será o mínimo entre os *xmax* dos 2 rectângulos... um raciocínio idêntico para as coordenadas *yy*... mas cuidado, isto só é válido quando há intersecção...

```
(define interseccao-rect
  (lambda (r1 r2)
```

Problema 3 **(5.0 valores)**

Pretende-se desenvolver um jogo em que o computador adivinha, em *n* perguntas (por exemplo 5), um número entre *0* e 2^n-1 pensado pelo utilizador (para *n=5*, é entre 0 e 31). Vamos supor que o número pensado foi 13.

```
> (adivinhar 5)   ; o argumento 5 significa que irão ser feitas 5 perguntas...

Pense num número entre 0 e 31!

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
P1- O numero pensado esta' no conjunto? s

2 3 6 7 10 11 14 15 18 19 22 23 26 27 30 31
P2- O numero pensado esta' no conjunto? n

4 5 6 7 12 13 14 15 20 21 22 23 28 29 30 31
P3- O numero pensado esta' no conjunto? s

8 9 10 11 12 13 14 15 24 25 26 27 28 29 30 31
P4- O numero pensado esta' no conjunto? s

16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
P5- O numero pensado esta' no conjunto? n

O numero pensado foi: 13
```

números entre 0 e 31 que, na sua representação na base 2, têm o bit menos significativo (bit 0) a 1

números que têm o bit 1 a 1

números que têm o bit 2 a 1

números que têm o bit 3 a 1

números que têm o bit 4 a 1

Para determinar o número pensado pelo utilizador, basta considerar que uma resposta ***n*** *(não)* vale zero e uma resposta ***s*** *(sim)*, em cada uma das perguntas, vale:

P1 - **1** P2 - **2** P3 - **4** P4 - **8** P5 - **16**

Resta depois somar os valores das várias respostas.
No exemplo apresentado, o número pensado é: 1 + 0 + 4 + 8 + 0 = 13.

3.1- Escrever APENAS OS NOMES e os PARÂMETROS dos principais procedimentos em que seria organizado o programa, e explique o objectivo de cada um deles.

3.2- Escrever o procedimento que corresponde ao corpo principal do programa *(resume-se, praticamente, à chamada dos principais procedimentos em que o programa se organiza).*

3.3- Escrever a parte do programa, acompanhado de eventuais procedimentos auxiliares, correspondente ao cálculo do número pensado pelo utilizador (com base nas respostas já coleccionadas por exemplo numa lista) e à visualização da mensagem final. No exemplo indicado, a referida mensagem é: **`O numero pensado foi: 13`**

(Fim.)