Licenciatura em Engenharia Informática (FCT/UNL)
Ano Lectivo 2008/2009
Linguagens e Ambientes Programação – Época Normal

15 de Junho de 2009 às 09:00

Exame com consulta com 2 horas e 45 minutos de duração + 15 minutos de tolerância

Nome: Num:

Notas: *Este enunciado é constituído por 6 perguntas e 7 folhas. Responda no próprio enunciado.*
Nos problemas em ML mostre que sabe usar o método indutivo.
Pode definir funções/métodos auxiliares sempre que precisar (muitas vezes é mesmo preciso)
Fraude implica reprovação na cadeira.

1. [2 valores] Dizem-se ESTÁTICAS, propriedades que podem ser completamente tratadas em tempo de compilação, portanto antes do programa começar a correr. Dizem-se DINÂMICAS, propriedades que têm de ser completamente tratadas em tempo de execução, portanto depois do programa começar a correr. Dizem-se MISTAS, propriedades que podem começar a ser tratadas em tempo de compilação, mas que têm de acabar de ser tratadas em tempo de execução.

   Para responder, no início de cada pergunta escreva na zona sublinhada uma das seguintes letras: **E** como abreviatura de ESTÁTICA, **D** como abreviatura de DINÂMICA, **M** como abreviatura de MISTA e **T** para indicar que todas as hipóteses são possíveis.

   _____ Em OCaml, a propriedade "no programa X existem acessos a variáveis intermédias", pode ser verificada estaticamente, ou tem de ser verificada dinamicamente?

   _____ Em OCaml e C, o endereço das variáveis locais pode ser determinado estaticamente, ou tem de ser determinado dinamicamente?

   _____ Em OCaml, só é válido aplicar o operador `&&` a expressões booleanas. Esta restrição pode ser verificada estaticamente, ou tem de ser verificada dinamicamente?

   _____ Em C, a situação do denominador duma divisão ser zero é absurda. No caso geral, esta restrição pode ser verificada estaticamente, ou tem de ser verificada dinamicamente?

   _____ A generalidade das linguagens de scripting dispõe de sistemas de tipos estáticos ou dinâmicos?

   _____ O sistema de tipos da linguagem Java é estático ou dinâmico?

   _____ Uma linguagem implementada através dum interpretador, tem um sistema de tipos estático ou dinâmico?

   _____ Numa linguagem com escopo dinâmico e com declaração do tipo das variáveis, a validação das atribuições (para verificar se as variáveis recebem valores de tipos compatíveis) pode ser feita estaticamente, ou tem de ser feita dinamicamente?

   _____ Em JavaScript, a determinação da *boa forma* dos literais da linguagem, por exemplo a verificação se as strings são correctamente delimitadas por um par de aspas, pode ser decidida estaticamente ou tem de ser decidida dinamicamente?

   _____ Em JavaScript, o operador de soma pode ser aplicado a valores que quase todos os tipos, e.g. `"ola"+1+1.5`, porque são permitidas conversões automáticas de tipo. Essas conversões são decididas estaticamente ou dinamicamente?

2. [2 valores] Diga qual o tipo da seguinte função em ML:

```
let f a b = b (a b) ;;
```

3. Horários de comboios em OCaml. Considere o seguinte horário parcial da Linha do Norte, representado por uma lista de percursos:

```
let timeTable = [
  [("Lisboa", 920); ("Vila Franca", 935); ("Santarém", 955); ("Coimbra", 1025); ("Porto", 1300)];
  [("Lisboa", 700); ("Coimbra", 800); ("Porto", 900)];
  [("Porto", 715); ("Coimbra", 835); ("Lisboa",1000)]
] ;;
```

Cada percurso é uma lista de pares (*estação*, *tempo*), indicando que um comboio, no seu trajecto, se imobiliza na estação *estação* no momento *tempo*. Um tempo é um inteiro que codifica horas e minutos de acordo com o seguinte exemplo: "920" significa "nove horas e vinte minutos".

Para representar horários de comboios, usamos os seguintes tipos:

```
type route = (string * int) list  ;;
type timetable = route list  ;;
```

a) [1.5 valores] Escreva em OCaml uma função

```
sufix : string -> route -> route
```

que, dada uma estação e um percurso, produza a porção desse percurso que começa na estação dada e termina no final desse percurso. Se a estação não ocorrer no percurso, devolva a lista vazia. Exemplos:

```
sufix "Santarém" [("Porto", 715); ("Coimbra", 835); ("Lisboa",1000)] = []
sufix "Coimbra" [("Porto", 715); ("Coimbra", 835); ("Lisboa",1000)] =
                                                  [("Coimbra", 835); ("Lisboa",1000)]
```

b) [1.5 valores] Escreva em OCaml uma função

```
prefix : string -> route -> route
```

que, dada uma estação e um percurso, produza a porção desse percurso que começa no seu início e termina na estação dada. Se a estação não ocorrer no percurso, devolver a lista vazia. Exemplos:

```
prefix "Santarém" [("Porto", 715); ("Coimbra", 835); ("Lisboa",1000)] = []
prefix "Coimbra" [("Porto", 715); ("Coimbra", 835); ("Lisboa",1000)] =
                                                  [("Porto", 715); ("Coimbra", 835)]
```

c) [2 valores] Escreva em OCaml uma função

```
direct : string -> string -> timetable -> route
```

que, dadas duas estações e um horário de comboios, descubra a porção de percurso mais directa (i.e. com menos estação intermédias) entre as estações dadas. Considera-se que **não pode haver mudança de comboio a meio da viagem**. A duração da viagem não interessa. Em caso de empate, produz-se qualquer uma das respostas possíveis. Se não existir ligação entre as duas cidades, devolve-se a lista vazia. Exemplos:

```
direct "Lisboa" "Porto" timeTable = [("Lisboa", 700); ("Coimbra", 800); ("Porto", 900)]
direct "Lisboa " "Coimbra " timeTable = [("Lisboa", 700); ("Coimbra", 800)]
direct "Santarém " "Coimbra " timeTable = [("Santarem", 955); ("Coimbra", 1025)]
direct "Lisboa" "Conímbriga" timeTable = []
```

[Sugestão: Use as funções das alíneas anteriores como auxiliares. Use também a função `List.length`. É possível escrever uma função relativamente simples e de tipo 1, para resolver este problema.]

d) [2 valores] Escreva em OCaml uma função

```
best : string -> string -> timetable -> route
```

que, resolve um problema semelhante ao da alínea c), **mas permitindo que possa haver mudanças de percurso a meio da viagem**. Agora minimiza-se em primeiro lugar o número de transbordos e em segundo lugar o número de estações intermédias. A duração da viagem não interessa. Se não existir ligação entre as duas cidades, devolve-se a lista vazia. Pode haver ciclos no sistema ferroviário, mas sabe-se que 50 é o número máximo de estações intermédias entre quaisquer duas estações. [Sugestão: Use a função da alínea anterior como auxiliar. Não se preocupe com a eficiência.]

4. [2 valores] Considere o seguinte programa escrito em GCC, uma variante do C que suporta aninhamento de funções:

```
int i = 8, j = 9 ;

void X(int a) {
    int i = 0 ;
    void Y(int j) {
          X(a + i + (j++)) ;
    }
    Y(a + i + (j++)) ;
}

int main(void) {
    X(0) ;
    return 0 ;
}
```

Mostre qual o estado da pilha de execução no momento em que nela se encontrem exactamente 6 registos de activação (incluindo os registos das funções `start` e `main`.)

Use as convenções habituais das aulas: Para efeitos da criação do registo de activação inicial, imagine que cada programa em GCC está embebido numa função sem argumentos chamada `start`. Depois trate todas as entidades globais do programa como sendo locais à função imaginária `start`. Assuma também que a primeira célula da pilha de execução é identificada como posição 00, a segunda célula da pilha de execução é identificada como posição 01, etc.

41	27	13
40	26	12
39	25	11
38	24	10
37	23	09
36	22	08
35	21	07
34	20	06
33	19	05
32	18	04
31	17	03
30	16	02
29	15	01
28	14	00

5. [3 valores] Considere o seguinte programa escrito em Ansi-C:

```
#include <stdio.h>
#include <setjmp.h>

#define MAX_NODE    10

typedef struct Node {
    int data;
    struct Node *next;
} Node, *List;

void A(void) {
    int i, a[MAX_NODE];
    int *p = a;

    for(i = 0; i < MAX_NODE; i++)
        a[i] = i;
    while( p++ - a < MAX_NODE - 1 )
        *p = *(p-1) ;
    for( i = 0; i < MAX_NODE; i++ )
        printf("%d ", a[i]) ;
    printf("\n") ;
}

void B(void) {
    static int b = 3;
    jmp_buf buf; // aula teórica 16

    if( setjmp(buf) != 0 )
        printf("%d\n", b);
    b = 5;
    longjmp(buf, 1);
}
```

```
void C(void) {
    Node n[MAX_NODE];
    List p;
    int i;

    n[MAX_NODE - 1].next = NULL;
    for( i = 0; i < MAX_NODE; i++ ) {
        n[i].data = i;
        n[i].next = &n[(i+1) % MAX_NODE];
    }
    for( p = n ; p != NULL ; p = p->next )
        printf("%d ", p->data);
    printf("\n") ;
}

int main(void) {
    A() ;
    B() ;
    C() ;
    return 0 ;
}
```

a) [1 valor] Diga qual o efeito produzido pela função **A**. Responda usando uma cruz X.

Imprime os dígitos de 0 a 9 e termina	____
Imprime os dígitos de 0 a 9 repetidamente e não termina	____
Imprime o dígito 0 dez vezes e termina	____
Imprime o dígito 0 nove vezes, depois um 9 e termina	____
Nenhuma das anteriores	____

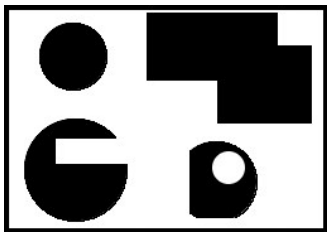
b) [1 valor] Diga qual o efeito produzido pela função **B**. Responda usando uma cruz X.

Imprime 3 e termina	____
Imprime 5 e termina	____
Imprime 3, depois 5, depois 3, depois 5, sem nunca terminar	__
Imprime 3 e depois uma sequência infinita de 5	____
Nenhuma das anteriores	____

c) [1 valor] Diga qual o efeito produzido pela função **C**. Responda usando uma cruz X.

Imprime os dígitos de 0 a 9 e termina	____
Imprime os dígitos de 0 a 9 repetidamente e não termina	____
Imprime o dígito 0 dez vezes e termina	____
Estoira com segmentation fault ao fim de 10 iterações	____
Nenhuma das anteriores	____

6. [4 valores] Imagens em C++. Analise a seguinte imagem bidimensional, desenhada a preto sobre fundo branco:



Trata-se duma imagem geométrica complexa que agrupa diversas sub-imagens. Analisemos algumas das sub-imagens: em cima à esquerda temos uma imagem básica, um simples círculo; baixo à esquerda temos uma imagem composta que resulta da subtracção dum rectângulo a um círculo.

O objectivo deste problema é a definição dum sistema de classes abstractas e concretas para a representação destas imagens.

Vamos considerar duas grandes categorias de imagens: **imagens básicas** - *círculo*, *rectângulo* e *quadrado* -, e **imagens compostas binárias** – *união* e *diferença*. Relativamente às imagens básicas: um círculo caracteriza-se pelo seu centro (um ponto) e pelo seu raio (um número real); um rectângulo (horizontal) caracteriza-se pelo seu topo esquerdo (um ponto) e pela sua base direita (outro ponto); um quadrado é um rectângulo especial. Quanto às imagens compostas binárias: cada uma delas caracteriza-se pelas duas sub-imagens constituintes.

***Problema*:** Escreva em C++ classes abstractas e concretas para representar imagens. O tipo mais geral deve chamar-se **Image**. Para além dos construtores, defina apenas a seguinte função pública, aplicável a todas as imagens:

```
bool Belongs(Point p) ;
```

A função `Belongs` testa se um ponto pertence a uma imagem, ou seja se pertence à sua parte preta. Os pontos na fronteira também fazem parte das imagens.

Para ajudar, assuma que já está disponível esta classe `Point`, pronta a usar:

```
class Point {
 private: double xval, yval ;
 public:  Point(double x, double y);
          double x() const ;
          double y() const ;
          double dist(Point other) const ;
}
```

[Nota: Este é um problema de modelação semelhante a diversos problemas vistos nas aulas: o problema das expressões algébricas, o problema das sucessões matemáticas, o problema das criaturas.]