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

06 de Julho 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 5 perguntas e 8 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] Perguntas de escolha múltipla. Responda aqui:

A	B	C	D	E	F

A) Na passagem de funções por parâmetro numa linguagem com aninhamento de funções:

a) Basta passar o apontador para o código da função.
b) Basta passar o apontador para o código da função e os valores para os seus argumentos.
c) É necessário passar o apontador para o código e também o static link da função.
d) Só é possível passar funções declaradas globalmente.

B) No seguinte código C++

```
class A {
  public: void m() {printf("A");}
};
class B : public A {
  public: void m() {printf("B");}
};
int main () {
   B b;
   A* a = &b;
   a->m();
   b.m();
}
```

o resultado impresso na consola é:

a) "AA" b) "BA" c) "AB" d) "BB"

C) No seguinte código C++

```
class A {
  public: virtual void m() {printf("A");}
};
class B : public A {
  public: virtual void m() {printf("B");}
};
int main () {
   B b;
   A* a = &b;
   a->m();
   b.m();
}
```

o resultado impresso na consola é:

a) "AA" b) "BA" c) "AB" d) "BB"

D) Em JavaScript é possível definir um array através do comando "`var a=[]`". Qual das situações é verdadeira:

a) O interpretador inicializa todas as posições do array com 0.
b) O comando "`a[10] = 0`" dá um erro de execução.
c) Qualquer acesso a uma posição não inicializada dá um valor especial "`undefined`".
d) Qualquer acesso a uma posição não inicializada do array dá um erro de execução.

E) O Polimorfismo divide-se em várias categorias. Qual destas situações se refere a polimorfismo paramétrico:

a)
```
let rec member x l = match l with
        [] -> false
      | y::l -> if x = y then true else member x l
    ;;
```
b)
```
class A : public B {...};
```
c)
```
float f = 2;
```
d) Nenhuma das anteriores.

F) Os sistemas de tipos nas linguagens de programação têm por missão evitar erros de execução. Nas linguagens tipificadas estaticamente que erros de execução é que, em geral, não são evitados?

a) A chamada de funções com um número errado de argumentos.
b) A utilização de um apontador com valor nulo.
c) A utilização de valores de tipos básicos diferentes numa mesma expressão (e.g. inteiros e strings).
d) Nenhum dos anteriores.

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

```
let rec f l n =
    match l with
       [] -> []
     | x::xs -> (x,x=n)::f xs n
;;
```

3.1. Considere a lista das notas finais obtidas por todos os alunos de Informática durante o ano lectivo de 2008/2009:

```
let grades0809 = [(3424,"LAP",20); (2105,"AC",19); (3424,"AM",15); (2222,"ASC",15); …] ;;
```

Cada elemento desta lista é um triplo ordenado constituído por: número de aluno, nome de disciplina, nota obtida. Para representar listas de notas, usamos o seguinte tipo:

```
type grades = (int * string * int) list ;;
```

a) [2 valores] Escreva em OCaml uma função

```
courses : grades -> string list
```

que produza a lista, sem repetições, dos nomes de todas as disciplinas que ocorrem numa lista de notas. A ordem do resultado não interessa. Exemplos:

```
courses [] = []
courses grades0809 = ["AM"; "LAP"; "AC"; …]
```

[Ajuda: Use a função predefinida `List.mem: a->a list->bool` para testar se um valor ocorre numa lista.]

b) [2 valores] Escreva em OCaml uma função

```
averages : grades -> (string * float) list
```

que produza a lista das médias das notas de cada disciplina, dada uma lista de notas finais. A ordem do resultado não interessa. Exemplos:

```
averages [] = []
averages grades0809 = [("LAP", 13.5); ("AC", 12.4); ("AM", 14.4); …]
```

[Ajuda:. Poderá interessar usar a função da alínea anterior e não se esqueça que pode sempre definir mais funções auxiliares Além disso, use a função predefinida `float_of_int: int->float` para converter inteiros em reais.]

3.2. É possível representar as precedências entre as diversas cadeiras da LEI usando uma árvore n-ária, a que vamos chamar **árvore de precedências**: as precedências de cada disciplina D estão agrupadas numa subárvore que tem a disciplina D como raiz. A disciplina "Projecto Integrador" fica na raiz da árvore geral porque depende, directa ou indirectamente, de todas as outras disciplinas.

Repare que uma disciplina pode ocorrer várias vezes na árvore de precedências, porque pode ser precedência de várias disciplinas ao mesmo tempo; no entanto, para evitar repetições desnecessárias, estabelecemos a seguinte regra: a subárvore completa de cada disciplina só aparece uma vez e todas as outras ocorrências dessa mesma disciplina aparecem como nós simples (sem filhos).

Para simplificar os problemas que se seguem, **vamos assumir que a árvore de precedências é simplesmente binária**. Em conformidade, assuma que o tipo duma árvore de precedências é o seguinte:

```
type predecences = Nil | Node of string * predecences * predecences ;;
```

Para exemplificar, se a LEI tivesse apenas as cadeiras "IP", "AED", "LAP" e "PI", a árvore de precedências seria:

```
let prec0809 = Node("PI", Node("AED", Node("IP",Nil,Nil), Nil), Node("LAP", Node("IP",Nil,Nil), Nil))
```

c) [1.5 valores] Escreva em OCaml uma função

```
prec : string -> predecences -> string list
```

que, dado o nome duma disciplina e uma árvore de precedências válida, produza a lista de todas as precedências directas ou indirectas dessa disciplina. A ordem do resultado não interessa e também não faz mal se ficarem repetições no resultado. Exemplos:

```
prec "ADA" prec0809 = []
prec "PI" prec0809 = ["LAP";"AED";"IP"]
prec "PI" prec0809 = ["LAP";"IP";"AED";"IP"]
prec "LAP" prec0809 = ["IP"]
```

d) [1.5 valores] Escreva em OCaml uma função

```
valid : predecences -> bool
```

para validar uma árvore de precedências. Uma árvore de precedências válida tem as duas seguintes propriedades: a) a subárvore completa de cada disciplina só aparece uma vez e todas as outras ocorrências dessa disciplina aparecem como nós simples (sem filhos); b) uma disciplina não pode ser precedência, directa ou indirecta, de si própria. Exemplos:

```
valid prec0809 = true
valid Nil = true
valid Node("PI", Nil, Node("AED", Node("PI",Nil,Nil), Nil)) = false
valid Node("PI", Node("AED", Node("IP",Nil,Nil), Nil), Node("AED", Node("IP",Nil,Nil), Nil)) = false
```

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

```
int *pt = NULL ;
void A(void) {
    void B1(void) {
        void C1(void) {
            int i = 936 ;
            pt = &i ;
            A();
        }
        C1();
    }
    void B2(void) {
        void C2(void) {
            B1();
        }
        C2();
    }
    B2();
}

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

Mostre qual o estado da pilha de execução no momento em que nela se encontrem exactamente 8 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.

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

5. [7 valores] Esta pergunta mistura a criação duma hierarquia de classes abstractas e concretas (típica do C++) com algumas manipulações de baixo nível (típicas do C).

   Um sistema de manufactura de uma fábrica de **circuitos impressos** recebe através duma conexão de rede a especificação para a construção dos circuitos impressos que fabrica. Trata-se de um sistema robotizado simples que usa placas de circuito impresso já preparadas e monta e solda cada um dos componentes no lugar certo.

   Circuito: Um circuito é constituído por uma a **placa base** (é identificada por um código - uma string de 10 caracteres) mais diversos **componentes electrónicos** agarrados (soldados) à placa base. Cada componente electrónico possui **dois ou mais terminais** através dos quais fica ligado à placa base.

   Componentes electrónicos: Cada componente electrónico de um circuito tem uma determinada posição e orientação na placa base, dadas por coordenadas `XY` na placa para cada um dos terminais do componente. Os componentes dividem-se em 4 categorias. Cada categoria tem um código associado e alguns atributos:

   - Resistência (Resistor) – Código 'R'. Atributos: valor de resistência eléctrica (int), valor de potência máxima (int), duas posições dos terminais (dois pares de inteiros).

   - Condensador (Capacitor) - Código 'C'. Atributos: valor (int), tipo cerâmico ou electrolítico ('C' ou 'E'), duas posições dos terminais (dois pares de inteiros).

   - Semicondutor vertical (Vertical semiconductor) – Código 'V'. Atributos: código de 6 caracteres (char[7]), número N de terminais (int), posição dos terminais (N pares de inteiros).

   - Semicondutor horizontal (Horizontal semiconductor) – Código 'H'. Atributos: código de 6 caracteres (char[7]), número N de terminais (int), posição dos terminais (N pares de inteiros), pode ter base amovível ou não (bool).

   Como se pode ver há **componentes elementares**, sempre apenas com dois terminais, e há **semicondutores**, cujo número de terminais pode ser grande (considere que o número máximo é 20).

   Representação física dos dados: A transmissão de dados é feita entre dois sistemas programados em C++ e por isso usa-se um esquema binário para representar os dados. Como certamente já sabe, a linguagem de programação Java dispõe de um mecanismo integrado que permite a leitura e escrita de um objecto (ou conjunto de objectos) a partir de uma (ou para uma) sequência de bytes recebida a partir de (ou enviada para) um ficheiro binário ou uma conexão de rede. A ideia é replicar um mecanismo semelhante em C++ (mas mais simples).

   A especificação de um circuito é lida de uma sequência de bytes que começa pela referência da placa base a ser utilizada (10 bytes), depois um inteiro que indica o número de componentes que se vão seguir e depois uma sequência de descrições, uma para cada um dos componentes, segundo o seguinte formato: para cada componente, aparece o seu código à cabeça e depois o valor dos seus atributos, pela ordem e usando os tipos da descrição anterior.

   Problema: Programe uma classe para representar circuitos. Programe também uma hierarquia de classes para representar componentes electrónicos individuais. Dentro de cada classe só precisa de programar os construtores e os destrutores. Não se pede mais nenhuma função. Tente que todos os construtores dos componentes tenham um argumento de entrada/saída de tipo `PT*` (apontador para apontador; o tipo PT está definido mais abaixo), para aceder e fazer avançar a posição corrente do buffer onde se encontra a descrição de um circuito. Tente fazer com que cada construtor consuma os bytes de que necessita e deixe o apontador-argumento no endereço onde começa a descrição do próximo componente.

   Pretende-se uma solução modular e extensível para este problema.

   Ajuda: Considere o seguinte código no qual se pode inspirar para escrever o construtor da classe dos circuitos:

```
#define BOARD_CODE_SIZE 10
#define MAX_COMPONENTS 20

typedef char *PT ;

void readCircuit(PT buffer){
  char[BOARD_CODE_SIZE + 1] board;
  memcpy(board, buffer, BOARD_CODE_SIZE);
  board[BOARD_CODE_SIZE] = '\0';
  PT pt = buffer + BOARD_CODE_SIZE;

  Component *components[MAX_COMPONENTS];
  int N = *((int*)pt); pt += sizeof(int);
  for( int i = 0 ; i < N ; i++ )
    switch(*((char*)(pt++))) {
      case 'R': components[i] = new Resistor(&pt); break;
      ...
    }
  ...
}
```