MAC 0316/5754 — Conceitos de Linguagens de Programação

Primeiro Semestre de 2011

Primeira Prova — 29 de março de 2011

Nome do aluno: _____

Assinatura: _____

Nº USP: _____

Instruções:

1. Preencha o cabeçalho acima.
2. Não destaque as folhas deste caderno.
3. A prova tem 4 questões. Antes de começar a trabalhar verifique se o seu caderno de questões está completo.
4. A prova deve ser resolvida individualmente. Não é permitida a consulta a livros, apontamentos ou colegas.
5. Não é permitido o uso de folhas avulsas para rascunho.

Boa prova!

Questão	Valor	Nota
1	2,5	
2	3,0	
3	3,0	
4	1,5	
Total	10,0	

Questão 1 (1,0 + 1,5 pontos)

(a) Escreva uma função `emparelha`, que recebe duas listas `l1` e `l2` e devolve uma lista cujos elementos são listas de comprimento dois. Cada uma dessas listas de comprimento dois tem seu primeiro elemento obtido da lista `l1` e seu segundo elemento obtido da lista `l2`. Exemplos de uso da função `emparelha`:

```
> (emparelha '(1 2 3 4 5) '(a b c d e))
'((1 a) (2 b) (3 c) (4 d) (5 e))
>
> (emparelha '(1 2 3) '(a b c d e))
'((1 a) (2 b) (3 c))
>
> (emparelha '(1 2 3 4 5) '(a b))
'((1 a) (2 b))
```

Note que o valor da função `emparelha` é uma lista de comprimento igual ao da menor (a mais curta) das duas listas passadas como parâmetros à função.

(b) Reescreva a sua função `emparelha` de modo que a área de memória empregada pela pilha de execução tenha tamanho constante (em vez de ser proporcional ao comprimento da menor das listas que se deseja emparelhar). Sugestão: Crie uma função auxiliar, seguindo uma estratégia análoga à vista em classe para a função `map`, e faça a nova função `emparelha` chamar a função auxiliar.

Questão 2 (1,5 + 1,5 pontos)

(a) Escreva uma função `medias`, que recebe uma função `crit` e duas listas `l1` e `l2`. Cada uma dessas listas contém as notas de uma mesma turma numa prova. O parâmetro `crit` é uma função que implementa o critério de cálculo da média. A função `medias` devolve uma lista com as médias da turma calculadas conforme o critério `crit`. A primeira nota da lista de médias é o resultado da aplicação de `crit` à primeira nota da lista `l1` e à primeira nota da lista `l2`, a segunda nota da lista de médias é o resultado da aplicação de `crit` à segunda nota da lista `l1` e a segunda nota da lista `l2`, etc. Exemplo de uso da função `medias`:

```
> (medias (lambda (p1 p2) (/ (+ p1 p2) 2)) '(100 50 70 40) '(90 80 70 60))
'(95 65 70 50)
```

(b) Reescreva a função `medias`, agora usando a função `map` de Scheme e a função `emparelha` da questão 1.

Questão 3 (3,0 pontos)

Este analisador sintático (*parser*) para a linguagem AE foi extraído do PLAI:

```
;; parse: s-expression -> AE
(define (parse sexp)
  (cond
    [(number? sexp) (num sexp)]
    [(list? sexp)
      (case (first sexp)
        [(+) (add (parse (second sexp))
                  (parse (third sexp)))]
        [(-) (sub (parse (second sexp))
                  (parse (third sexp)))])]))
```

Usando a função acima como base, escreva um analisador sintático para a linguagem F1WAE. Esta é a definição, em BNF, da sintaxe concreta da linguagem F1WAE:

```
<F1WAE> ::= <num>
        | {+ <F1WAE> <F1WAE>}
        | {with {<id> <F1WAE>} <F1WAE>}
        | <id>
        | {<id> <F1WAE>}
```

Esta é a definição, em PLAI Scheme, da sintaxe abstrata da linguagem F1WAE:

```
(define-type F1WAE
  [num (n number?)]
  [add (lhs F1WAE?) (rhs F1WAE?)]
  [with (name symbol?) (named-expr F1WAE?) (body F1WAE?)]
  [id (name symbol?)]
  [app (fun-name symbol?) (arg F1WAE?)])
```

Seu analisador sintático não precisa fazer deteção e tratamento de erros sintáticos. Basta que ele funcione para programas sintaticamente corretos.

Questão 4 (1,5 pontos)

Este interpretador para a linguagem F1WAE usa substituição postergada:

```
;; interp: F1WAE listof(FunDef) DefrdSub-> number
(define (interp expr fun-defs ds)      ; Função extraída do PLAI
  (type-case F1WAE expr
             [num (n) n]
             [add (l r) (+ (interp l fun-defs ds)
                           (interp r fun-defs ds))]
             [with (bound-id named-expr bound-body)
                   (interp bound-body
                           fun-defs
                           (aSub bound-id
                                 (interp named-expr
                                         fun-defs
                                         ds)
                                 ds))] ; <-------------- (1) E se aqui tivéssemos (mtSub) em vez de ds?
             [id (v) (lookup v ds)]
             [app (fun-name arg-expr)
                  (local ([define the-fun-def (lookup-fundef fun-name fun-defs)])
                    (interp (fundef-body the-fun-def)
                            fun-defs
                            (aSub (fundef-arg-name the-fun-def)
                                  (interp arg-expr fun-defs ds)
                                  (mtSub))))])) ; <----- (2) E se aqui tivéssemos ds em vez de (mtSub)?
```

(a) Escreva uma expressão na linguagem F1WAE que mostre que o `ds` no ponto (1) não pode ser substituído por `(mtSub)`. Em outras palavras, escreva uma expressão na sintaxe concreta da linguagem F1WAE que (depois de traduzida para a sintaxe abstrata) não seja avaliada corretamente por uma versão da função `interp` com a modificação sugerida no ponto (1). Forneça também os valores dos argumentos `fun-defs` e `ds` que devem ser passados à função `interp` juntamente com a expressão.

(b) Escreva uma expressão na linguagem F1WAE que mostre que o `(mtSub)` no ponto (2) não pode ser substituído por `ds`. Em outras palavras, escreva uma expressão na sintaxe concreta da linguagem F1WAE que (depois de traduzida para a sintaxe abstrata) não seja avaliada corretamente por uma versão da função `interp` com a modificação sugerida no ponto (2). Forneça também os valores dos argumentos `fun-defs` e `ds` que devem ser passados à função `interp` juntamente com a expressão.

Rascunho (não destacar esta página do caderno de questões)