Lista de Linguagens de Programação – 13

Nome: _____ Matrícula: _____

1. Considere uma linguagem com estrutura por blocos, que use o link aninhado em chamada de funções, tal qual ML. Suponha que uma função aninhada em uma profundidade de n níveis faça uma referência (legítima) a uma variável definida no escopo de uma função aninhada em uma profundidade de m níveis. Descreva exatamente como encontrar a variável em tempo de execução. Por que não é necessário tratar de forma especial o caso em que $m > n$?

2. Escreva o mais curto exemplo, em SML/NJ em que você consiga pensar, que não funcionaria se implementado usando um registro de ativação alocado estaticamente. Explique porque este exemplo falharia.

3. Escreva o mais curto exemplo, em SML/NJ, em que você consiga pensar, que não funcionaria corretamente caso SML/NJ não implementasse registros de ativação usando links de aninhamento. Isto é, os registros de ativação podem ser implementados via uma pilha, como é normalmente feito, mas não possuem link de aninhamento.

4. Para cada função abaixo, escrita em SML/NJ, diga se o seu registro de ativação pode ser desalocado assim que a função retornar. Explique o porquê.

   (a) `fun f x = x + 1`

   (b) `fun f x = fn y => x + y`

   (c) `fun f x = fn y => y + 1`

   (d) `fun f x = map   x`

5. O objetivo desta questão é escrever uma função `concat` de várias formas diferentes. Esta função recebe como entrada uma lista de strings, e retorna uma única string, formada a partir da concatenação dos elementos da lista. Por exemplo, `concat ["ab", "cd", "ef"] = "abcdef"`.

   (a) Escreva a função `concat` usando recursão explícita. Neste caso, programe "indutivamente", isto é, defina um caso base, quando a lista de entrada estiver vazia, e defina um caso de indução. No passo indutivo, use o seguinte raciocínio para escrever o programa: dado que você sabe concatenar uma lista de n elementos, como fazer para concatenar uma lista de $n + 1$ elementos?

   (b) Escreva a função `concat` em uma linha, usando a função `foldr`.

6. Cada uma das funções abaixo, escrita em SML/NJ, contém uma chamada de função que passa um parâmetro `f`, de alta ordem. Em cada caso, diga se a função `f` usa o link de aninhamento quando chamada. Justifique a sua resposta.

   (a)
```
fun addone theList =
    let fun f x = x + 1
    in map f theList
    end
```

   (b)
```
fun addall n theList =
    let fun f x = x + n
    in map f theList
    end
```

   (c)
```
fun do123 f = map f [1, 2, 3]
```

7. Considere o seguinte trecho de uma seção de Python:

```
>>> x = 1
>>> def inc(y): return x + y
...
>>> print(inc(2))
3
>>> x = 2
>>> print(inc(2))
4
```

Com base nos resultados impressos, é possível concluir que Python possui escopo dinâmico? Justifique a sua resposta.

8. Vê-se abaixo uma implementação do algoritmo *quicksort*, na linguagem SML:

```
fun leq a b = a <= b

fun grt a b = a > b

fun filter _ nil = nil
  | filter f (h::t) = if f h then h :: filter f t else filter f t

fun qsort nil = nil
  | qsort (h::t) = (qsort (filter (grt h) t)) @ [h] @ (qsort (filter (leq h) t))
```

Nesse código *closures* estão sendo usados para criar pequenos programas. Por exemplo, a chamada `(grt h)` cria uma nova função. Essa função retorna verdadeiro sempre que ela recebe um número menor que o parâmetro `h`.

(a) A função `(grt h)` é um *closure*. Esse tipo de entidade possui uma representação muito natural no cálculo λ. Seja $grt = \lambda a.\lambda b.a > b$. Assumindo `h = 21`, o que é grt `21`?

9. É possível simular funções de alta ordem em C, passando-se endereços de procedimentos como parâmetros. Considere, por exemplo, a implementação abaixo:

```
#include <stdio.h>
#include <stdlib.h>
int* map (int* a, unsigned size, int (*f)(int)) {
  int i = 0;
  int* narray = (int*) malloc(size * sizeof(int));
  while (i < size) {
    narray[i] = (*f)(a[i]);
    i++;
  }
  return narray;
}
int inc(int x) { return x + 1; }
int sqr(int x) { return x * x; }
int main() {
  int a[] = {2, 3, 5, 7, 11, 13, 17, 19, 20};
  int* b;
  int* c;
  b = map(a, 9, inc);
  c = map(a, 9, sqr);
}
```

(a) Qual o conteúdo do arranjo `b` ao final do programa?

(b) E qual o conteúdo do arranjo `c` ao final do programa?

(c) A linguagem C fornece alguma sintaxe para que o valor de retorno de uma função seja outra função?

10. A linguagem ANSI C não fornece sintaxe para a declaração de funções aninhadas. Por outro lado, os dialetos de C compilados pelos compiladores mais populares, como `gcc` ou `clang` em geral suportam esse tipo de contrução. Por exemplo, é possível compilar o programa abaixo usando-se `gcc` [1]:

```
void testMap(int y) {
  int nestedInc(int x) { return x + y; };
  int a[] = {2, 3, 5, 7, 11, 13, 17, 19, 20};
  int* b;
  b = map(a, 9, nestedInc);
  printvec(b, 9); // Imprime as 9 primeiras casas do vetor.
}
int main(int argc, char** argv) {
  testMap(argc);
}
```

(a) Assumindo-se que a função `map` foi implementada como na questão anterior, e que o programa acima foi compilado em um binário `a.out`, o que será produzido pela chamada `./a.out a b c`?

(b) Procure saber: a linguagem C possui *closures*?

[1] Possivelmente será necessário passar parâmetros extra para o compilador, e.x.: `gcc -fnested-functions arq.c`.