Lista de Linguagens de Programação – 19

Nome: _____ Matrícula: _____

1. A passagem de parâmetros por expansão de macros é um mecanismo bastante utilizado em C. Um exemplo é dado abaixo:

```
#define SUM(X, Y) (X) + (Y)

int main(int argc, char** argv) {
  printf("sum = %d\n", SUM(argc, argv[0][0]));
}
```

   (a) Macros são expandidas por um componente do compilador chamado *pré-processador*. Escreva o código do programa acima, após o pré-processamento.

   (b) Um dos problemas com expansão de macros é a chamada *captura de variáveis*. Ilustre este problema com um exemplo.

   (c) Um outro problema é a múltipla avaliação de parâmetros, isto é, o fato de parâmetros serem avaliados múltiplas vezes pode dar ao programa um significado diferente da intenção do programador. Escreva um exemplo em C que prove que parâmetros de macros são avaliados múltiplas vezes.

2. Consider a classe abaixo, implementada em Java:

```
class MyInt {
  int i;
  MyInt(int k) {i = k;}
  void swap1(MyInt j) {
    MyInt tmp = j;
    j = new MyInt(i);
    i = tmp.i;
  }
  void swap2(MyInt j) {
    MyInt tmp = j;
    j.i = i;
    i = tmp.i;
  }
  void swap3(int j) {
    int tmp = j;
    j = i;
    i = tmp;
  }
}
```

Cada uma das próximas questões é completamente independente uma das outras. Estas questões devem ser respondidas com base nas definições abaixo:

```
MyInt m1 = new MyInt(3);
MyInt m2 = new MyInt(4);
```

(a) Qual é o valor de `m1.i` e `m2.i` depois da chamada `m1.swap1(m2)`?

(b) Qual é o valor de `m1.i` e `m2.i` depois da chamada `m1.swap2(m2)`?

(c) Qual é o valor de `m1.i` e `m2.i` depois da chamada `m1.swap3(m2)`?

(d) Qual é a política [1] que java adota para passar tipos primitivos (`int`, `float`, `char`, etc) como parâmetros de métodos?

(e) Qual é a política que java adota para passar objetos como parâmetros de métodos?

[1] As políticas existentes são passagem por: valor, referência, nome, expansão de matro, retorno, valor-retorno e avaliação preguiçosa

3. Neste exercício você criará uma class `MySet` em Python que representa um conjunto de dados. Existem muitas formas de se implementar esta classe, mas para este exercício a classe será implementada como uma árvore binária de busca. Lembra aquela lista em que você implementou uma árvore assim em SML? Pois é... A árvore não precisa ser balanceada. Além da classe `MySet`, você precisará criar mais algumas classes, para representar os nodos da árvore, por exemplo. A classe `MySet` deverá ter os seguintes componentes:

   - Um construtor, de modo que `MySet()` cria objetos que representam o conjunto vazio.
   - Um método `find` tal que `x.find(n)` retorne `True` se `n` estiver presente em `x`, e `False` caso contrário. Ainda bem que a nossa árvore é uma árvore binária de busca, não é?
   - Um método `add`, de tal forma que `x.add(n)` não retorne nenhum valor, mas tenha o efeito colateral de inserir `n` no conjunto `x`. Se `n` já estiver em `x`, então nada deve ser modificado.
   - Um método `__str__()`, tal que `x.__str__()` retorne uma *string* representando o conteúdo ordenado do conjunto `x`. Por exemplo, caso `x` represente o conjunto $\{1, 7, 2, 5\}$, então `x.__str__()` deveria imprimir "{1, 2, 5, 7}".

4. O objetivo deste exercício é entender um pouco melhor os mecanismos de passagem de parâmetros adotados em SML/NJ. Para isto, responda as duas questões abaixo:

   (a) Faça um experimento em ML que prove que esta linguagem não usa passagem de parâmetros por nome.

   (b) Escreva duas funções ML, f e g, que demonstrem que ML não faz passagem de parâmetros por expansão de macros. A função f deve chamar apenas a função g, e a função g não deve chamar nenhuma outra função. Explique os resultados que a chamada por expansão de macros produziria, e mostre os resultados que ML efetivamente produz.

5. Um dos mecanismos de passagem de parámetros chama-se passagem por nome. Na passagem por nome os parâmetros não são avaliados imediatamente. A avaliação acontece quando estes parâmetros são utilizados. Um detalhe importante, contudo, é que a avaliação acontece no contexto da chamada da função, e não no contexto da função chamada. Isto faz com que não ocorra o problema da captura de variáveis, como ocorre em expansão de macros. A idéia de passagem por nomes foi lançada em Algol, e viu uso também em Simula, aquela mesma linguagem que foi a precursora das linguagens orientadas por objeto. Por outro lado, este mecanismo acabou abandonado, pois era difícil de ser implementado, e complicava o código dos programas. É claro, entretanto, que podemos fazer coisas maravilhosas com a chamada por nome. Considere, por exemplo, a função abaixo, escrito em Simula, e responda as perguntas a seguir:

```
Real Procedure Sigma (k, m, n, u);
   Name k, u;
   Integer k, m, n; Real u;
Begin
   Real s:=0;
   k:= m;
   While k <= n Do Begin s:= s + u; k:= k + 1; End;
   Sigma:= s;
End;
```

(a) Qual o valor de `Z` na chamada abaixo?

```
Z:= Sigma (i, 1, 4, i ** 2);
```

(b) O que o programa abaixo faz? Não precisa escrever o valor calculado, somente explicar o que ele calcula:

```
a := io.read_integer();
Z := Sigma (i, 1, 100, 1 / (i + a) ** 2);
```