Lista de Exercícios de Linguagens de Programação I
Universidade Federal do Amazonas / Instituto de Computação
Marco Cristo

Introdução

1) Qual o principal objetivo da criação do Fortran, que contrapunha linguagens de primeira e segunda geração? Especificamente, qual o público-alvo do Fortran?

2) Embora muito pouco usada, o Algol teve enorme influência sobre projetos modernos. Cite características do Algol observadas em linguagens modernas.

3) Uma meta de projeto comum de C e Java foi a portabilidade. Muitas das características adotadas por C para atingi-la levaram à criação de uma linguagem potencialmente insegura (para se ter uma linguagem pequena e simples, optou-se por pouco rigor em checagem de tipos, o controle de recursos como memória foi deixado para o programador, etc). Que solução foi adotada por Java para o mesmo objetivo sem sacrificar segurança? O que foi sacrificado neste caso?

4) Qual a diferença entre linguagens de *script* e linguagens de finalidade geral?

5) Que linguagens introduziram os principais paradigmas de programação, em termos de prevalência no tempo e impacto sobre as linguagens futuras?

Aquecimento em Programação Funcional

6) Defina os seguintes conceitos como são compreendidos no paradigma funcional: (a) funções anônimas, (b) funções de primeira classe, (c) funções de alta ordem, (d) funções recursivas, (e) funções puras, (f) avaliação preguiçosa, (g) iteradores e (h) geradores ou *streams*.

7) Usando LCs, defina a função *pares*(*numero*) que retorna o número de dígitos pares em *numero*. Ex: `pares(42134513) = 3`. Dica: str(n) retorna n como uma string que, em Python, é uma lista de caracteres.

8) Defina a função *menos_que_k_vogais*(*string*) que retorna as palavras em *string* com menos que *k* vogais. Ex: `menos_que_k_vogais('este eh um teste', 2) = ['eh', 'um']`. Dica: string.split() retorna uma lista onde cada elemento é uma palavra de string.

9) Implemente a função *maioria* que tem como parâmetros uma propriedade (descrita como uma função anônima) e uma lista e retorna True se mais que metade dos elementos na lista têm a propriedade. Ex: `maioria(lambda x:x>1, [1, 2, 1, 3, 0, 2, 4]) = True`.

10) Implemente a função *composição* que tem como parâmetros uma lista de funções e um número e retorna o valor resultante da composição das funções aplicadas ao número. Ex: `composicao([], 4) = 4; composição([lambda n:n+2, lambda n:n*n], 3) = 25`. Obs: atenção para a ordem -- composição([f, g, h], n) = h(g(f(n))).

O jogo de Bacará

O jogo de bacará consiste em várias rodadas onde um jogador aposta no vencedor de cada rodada, ele ou a mesa. Ganha uma rodada quem obtém a mão de maior valor. O jogo inicia

com o *dealer* embaralhando 4 baralhos de 52 cartas. O jogador então aposta se ele irá vencer, perder (vitória da mesa ou banqueiro) ou empatar. O *dealer* inicialmente descarta dois pares de cartas, uma para o jogador, outro para a mesa. Se a mão do jogador tem menos de seis pontos, ele descarta outra carta. Dependendo dos pontos da mesa e do jogador, uma nova carta é descartada para a mesa. As mãos são então avaliadas, o vencedor é anunciado e são pagas as apostas. A partir desse ponto, uma nova rodada se inicia. A ideia geral do jogo de bacará pode ser descrito pelas seguintes funções em Python:

```
def bacara(deque):
	deque = deque_embaralhado(4 * deque)
	print 'Bem vindo ao jogo de bacara!'
	ptos = 0.0
	while len(deque) > 1:
		aposta = raw_input(
			'No que aposta (-1: mesa, 1: voce, 0: empate)? [-1] ')
		aposta = -1 if not aposta else int(aposta)
		pt, deque = rodada(deque, aposta)
		ptos += pt
		print 'Pontuacao rodada:', pt
		print '          global:', ptos
		resp = raw_input('Outra rodada? [s] ')
		if not (resp in ['s', 'S', '']):
			break

def rodada(deque, aposta):
	(deque, mjogador) = descartar(2, deque, [], len(deque) >= 2)
	(deque, mmesa) = descartar(2, deque, [], len(deque) >= 2)

	vmj = valor_mao(mjogador)
	vmm = valor_mao(mmesa)

	print '  Sua mao: [', lista(mjogador), '] =', vmj
	print '     Mesa: [', lista(mmesa), '] =', vmm

	(deque, mjogador) = descartar(1, deque, mjogador, len(deque) >= 1 and vmj < 6)
	vmj = valor_mao(mjogador)
	print ' Nova mao: [', lista(mjogador), '] =', vmj

	(deque, mmesa) = descartar(1, deque, mmesa,
		len(deque) >= 1 and tira_outra(vmm, vmj))
	vmm = valor_mao(mmesa)
	print 'Nova Mesa: [', lista(mmesa), '] =', vmm

	return pontos(aposta, vmj, vmm), deque
```

A função bacará supõe a existência de um deque de cartas, que será criado no exercício 6. Ela, em geral, é mais voltada para I/O, estando implementada basicamente de forma procedural. A função rodada retorna o número de pontos feito pelo jogador na rodada, dada sua aposta e o deque de cartas. Ela depende de uma série de outras funções que serão implementadas nos exercícios a seguir.

11) Dadas as listas `naipes = ['♣', '♦', '♥', '♠']` (paus, ouros, copas e espada) e `ranks = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']`, crie uma lista que represente um baralho com 52 cartas representadas como tuplas, usando uma LC. Ex: `[('2', '♣'), ('2', '♦'), ('2', '♥'), ('2', '♠'), ('3', '♣'), ('3', '♦'),..., ('A', '♣'), ('A', '♦'), ('A', '♥'), ('A', '♠')]`. Dica: para ter uma lista de naipes como a dada neste exercício, você deve usar símbolos Unicode: `naipes = [s.encode('utf-8') for s in [u"\u2660", u"\u2661", u"\u2662", u"\u2663"]]`.

12) Crie uma função que dada uma lista *lst* e um valor *k*, retorne *lst* sem o *k*-ésimo elemento. Ex: `sem_elemento_k([1,2,3], 1) = [1, 3]`.

13) Crie uma função que dada a lista, retorne uma versão embaralhada da mesma. Ex: `deque_embaralhado([1, 2, 3, 4, 5]) = [2, 3, 5, 1, 4]`, usando recursão e a função *sem_elemento_k*. DIca: você pode criar uma lista embaralhada usando uma lista auxiliar. Basta tirar um elemento aleatório da primeira e colocá-lo na segunda. A segunda lista será uma versão embaralhada da primeira. Para gerar números aleatórios, utilize a função *randint* do pacote *random*.

14) Uma mão é uma sequência de cartas. Escreva uma função que, dada uma sequência de cartas na forma de uma lista, produza uma string correspondente à lista de cartas. Exemplo: `lista[('J','♥'),('5','♣'),('10','♦')]) = 'J♥ 5♣ 10♦'`.

15) Escreva uma função para avaliar uma mão de cartas no jogo *bacará*. Neste jogo, as cartas são avaliadas como segue. 10, Rei (K), rainha (Q), valete (J) valem 0 pontos cada. As cartas 2 a 9 valem seus valores de face (ou seja, a carta 7 vale 7 pontos). A carta Ás (A) vale 1 ponto. O valor da mão corresponde à soma dos valores das cartas, descontada a dezena. Assim, caso a soma dos valores da mão seja 23, a mão é avaliada como 3. Exs: `valor_mao([('J','♥'), ('5','♣'), ('10','♦')]) = 5; valor_mao([('A','♥'), ('Q','♣')]) = 1; valor_mao([('A','♥'), ('2','♣'), ('A','♦')]) = 4;`

16) Descreva a função *descartar* que, dados um número de cartas *k*, um deque *dq*, uma mão e um critério de descarte, retorna *dq* sem suas *k* primeiras cartas e a mão acrescida das *k* primeiras cartas retiradas do deque, *se* o critério dado for verdadeiro. Caso contrário, são retornados o deque e a mão originais Ex: Considere `dq = [('6','♣'), ('8','♦'), ('A','♠'), ('6','♠')]`, então `descartar(2, dq, [('10','♠'), ('K','♣')], len(dq) >= 2) = ([('10','♠'), ('K','♣'), ('6','♣'), ('8','♦')], [ ('A','♠'), ('6','♠')])`.

17) No jogo de bacará, o jogador e a mesa (o banqueiro) só recebem até 3 cartas. A mesa, contudo, recebe sua terceira carta, dependendo do valor de sua mão (*vmm*) e do valor da mão do jogador (*vmj*), de acordo com a tabela abaixo. Nesta tabela, N significa que a mesa não recebe uma terceira carta enquanto S significa que ela recebe. Escreva a função *tira_outra*(*vmm*, *vmj*) que retorna True se a mesa pode descartar uma terceira carta. Ex: `tira_outra(7, 3) = False`; `tira_outra(6, 7) = True`;

	vmj									
vmm	0	1	2	3	4	5	6	7	8	9
9	N	N	N	N	N	N	N	N	N	N
8	N	N	N	N	N	N	N	N	N	N
7	N	N	N	N	N	N	N	N	N	N
6	N	N	N	N	N	N	S	S	N	N
5	N	N	N	N	S	S	S	S	N	N
4	N	N	S	S	S	S	S	S	N	N
3	S	S	S	S	S	S	S	S	N	S
2	S	S	S	S	S	S	S	S	S	S
1	S	S	S	S	S	S	S	S	S	S
0	S	S	S	S	S	S	S	S	S	S

18) Ganha uma rodada do jogo de bacará quem possui a mão de maior valor. O número de pontos feitos pelo jogador, contudo, depende da aposta feita no início da rodada. Se ele

erra o vencedor, ele perde 1 ponto. Se ele apostou na sua própria vitória e acertou, ele faz um ponto; se apostou na vitória da mesa, ele faz 0.95 pontos; se apostou em empate, ele faz 8 pontos. Descreva a função *pontos*(*aposta, vmj, vmm*) que, dado a aposta do jogador (-1 para vitória da mesa, 0 para empate e +1 para sua vitória), o valor da sua mão *vmj* e o valor da mão da mesa *vmm*, ela retorna o número de pontos do jogador. Ex: `pontos(-1, 7, 8) = 0.95; pontos(1, 7, 8) = -1.0; pontos(0, 7, 8) = -1.0; pontos(0, 7, 7) = 8.0.`

19) Baseado em sua experiência com Python Funcional, a compare com C, considerando os seguintes critérios:

    a. Simplicidade, Ortogonalidade, Tipos de Dados, Projeto de Sintaxe, Suporte à Abstração, Expressividade, Checagem de Tipos, Manipulação de Exceções e Restrição de *Aliases*
    b. Legibilidade, escrita e confiabilidade

Observações

- Resposta pode ser entregue em dupla;
- Plágio não será tolerado, com anulação dos casos observados;
- Data de entrega a ser definida no site da disciplina;
- Além da lista, devem ser enviado o código fonte completo correspondente ao jogo de bacará, junto com instruções de como testá-lo;
- A parte prática é a de maior peso nesta lista;