Lista de Exercícios 1

Teoria da Computação — Mestrado em Computação Aplicada

Prof. Jefferson O. Andrade

Ifes — Campus Serra — PPComp

2022/2

1 Introdução

Resolva os exercícios abaixo, e gere um relatório de resolução em PDF. Utilize a classe `scrartcl` do latex para produzir o seu relatório, com as definições padrão de fonte em tamanho 12 e papel tamanho A4.

O trabalho deve ser apropriadamente identificado e caso haja a necessidade de usar notação matemática, esta deve estar apropriadamente formatada.

O trabalho é individual.

2 Exercícios

Exercício 1

Implemente cada um dos programas a seguir como um programa SISO Python. Em cada caso, você pode assumir que a entrada é válida (ou seja, consiste em uma lista de números inteiros formatada corretamente).

(a) Escreva um programa que receba como entrada uma lista de inteiros separados por espaço em branco. A saída é uma string que representa a soma de cada segundo inteiro na lista. Por exemplo, se a entrada for "58 41 78 3 25 9", a saída será "53", porque $41 + 3 + 9 = 53$.

(b) Escreva um programa, semelhante ao programa em (a), mas somando cada terceiro elemento da entrada em vez de cada segundo elemento.

(c) Escreva um programa de decisão que aceite uma lista de inteiros se a soma de cada terceiro elemento for maior que a soma de cada segundo elemento, e rejeite caso contrário. Seu programa deve importar e usar os programas de (a) e (b).

Exercício 2

Considere o programa `oooops.py` dado abaixo.

```
def oooops(inp):
    try:
        val = int(inp)
    except ValueError:
        val = len(inp)
    s = 'A'
    i = 0
    while i != val:
        s += inp[2]
        i += 1
    return s
```

Responda às seguintes perguntas:

(a) Qual o resultado de `oooops("abc")`?

(b) Qual o resultado de `oooops("abcdefghij")`?

(c) Qual o resultado de `oooops("a")`?

(d) Qual o resultado de `oooops("008")`?

(e) Qual o resultado de `oooops("8")`?

(f) Qual o resultado de `oooops("-11")`?

(g) Descreva o conjunto de todas as strings I para as quais `oooops(I)` é indefinido.[1]

Exercício 3

Use a prova por contradição para provar as seguintes afirmações:

(a) Existem infinitos números inteiros positivos.

(b) Existem infinitos números inteiros negativos.

(c) Existem infinitos números pares.

(d) Não existe o menor número real positivo.

Exercício 4

Como indicado na Figura 3.3 do livro "What Can Be Computed?", a saída do comando `containsGAGA(rf('containsGAGA.py'))` é "yes". Escreva uma nova versão deste programa, chamada `containsGA_GA.py`. Essa nova versão deve ser equivalente à antiga, i.e., produzir as mesmas respostas para as mesmas entradas. Além disso os comandos `containsGA_GA(rf('containsGA_GA.py'))` e `containsGAGA(rf('containsGA_GA.py'))` devem ambos retornar "no".

[1] De acordo com a definição de programa dada na seção 2.4 do livro "What Can Be Computed?".

Exercício 5

Considere os programas `yesOnStringApprox`, `yesOnSelfApprox`, e `notYesOnSelfApprox` abaixo.

```
# SISO program yesOnStringApprox.py

# This program approximates the desired behavior of yesOnString.  It
# works correctly for four particular values of progString, but
# returns "unknown" for all other values of progString. See the
# exercises of chapter 3 for details.
import utils; from utils import rf
from containsGAGA import *
from longerThan1K import *
from maybeLoop import *
from yes import *
def yesOnStringApprox(progString, inString):
    if progString == rf('containsGAGA.py'):
        return containsGAGA(inString)
    elif progString == rf('longerThan1K.py'):
        return longerThan1K(inString)
    elif progString == rf('yes.py'):
        return yes(inString)
    elif progString == rf('maybeLoop.py'):
        if not 'secret sauce' in inString:
            return 'no'
        else:
            return maybeLoop(inString)
    else:
        return 'unknown'




def testyesOnStringApprox():
    testvals = [
        ('containsGAGA.py', 'TTTTGAGATT', 'yes'),
        ('containsGAGA.py', 'TTTTGAGTT', 'no'),
        ('longerThan1K.py', 1500*'x', 'yes'),
        ('longerThan1K.py', 'xyz', 'no'),
        ('yes.py', 'xyz', 'yes'),
        ('maybeLoop.py', '', 'no'),
        ('maybeLoop.py', 'asdfhjksd', 'no'),
        ('maybeLoop.py', 'secret sauce', 'yes'),
        ('maybeLoop.py', 'xsecret sauce', 'no'),
        ('maybeLoop.py', 'xsecret saucex', 'yes'),
    ]
    for (filename, inString, solution) in testvals:
        val = yesOnStringApprox(rf(filename), inString)
        utils.tprint(filename + ":", val)
        assert val == solution

```

```
# SISO program yesOnSelfApprox.py

# This program approximates the desired behavior of yesOnSelf.  It
# works correctly on certain values of progString. See the exercises
```

```
# of chapter 3 for details.
import utils; from utils import rf
from yesOnStringApprox import yesOnStringApprox
def yesOnSelfApprox(progString):
    return yesOnStringApprox(progString, progString)


def testyesOnSelfApprox():
    testvals = [
        ('containsGAGA.py', 'yes'),
        ('longerThan1K.py', 'no'),
        ('yes.py', 'yes'),
    ]
    for (filename, solution) in testvals:
        val = yesOnSelfApprox(rf(filename))
        utils.tprint(filename + ":", val)
        assert val == solution

```

```
# SISO program notYesOnSelfApprox.py

# This program approximates the desired behavior of notYesOnSelf.  It
# works correctly on certain values of progString.
import utils; from utils import rf
from yesOnSelfApprox import yesOnSelfApprox
def notYesOnSelfApprox(progString):
    val = yesOnSelfApprox(progString)
    if (val == 'yes'):
        return 'no'
    elif (val == 'no'):
        return 'yes'
    else:
        return 'unknown'


def testnotYesOnSelfApprox():
    testvals = [
        ('containsGAGA.py', 'no'),
        ('longerThan1K.py', 'yes'),
        ('yes.py', 'no'),
    ]
    for (filename, solution) in testvals:
        val = notYesOnSelfApprox(rf(filename))
        utils.tprint(filename + ":", val)
        assert val == solution

```

Após estudar os programas acima, determine a saída dos seguintes comando em Python:

(a) `yesOnStringApprox(rf('longerThan1K.py'), rf('longerThan1K.py'))`

(b) `yesOnStringApprox(rf('maybeLoop.py'), rf('maybeLoop.py'))`

(c) `yesOnSelfApprox(rf('longerThan1K.py'))`

(d) `notYesOnSelfApprox(rf('containsGAGA.py'))`