22c:016 Computer Science I: Foundations
Exam 1 Solutions
Thursday, September 25, 2014

1. Expressions (50 points)

Evaluate each of the following expressions. If an expression cannot be evaluated (*i.e.*, it yields an error) explain why. Moreover, assume the following variables have the specified values:

L = [’taken’,’surprise’,’over’,’come’,’by’]
s = ’coffee coke tea lemonade milk sprite’

(1) 175//2.0

Answer: 87.0 Note that the type is float, even though it is "floor" division.

(2) "spring" + ("50")

Answer: ’spring50’

(3) "100" == 100

Answer: False Types don’t match!

(4) (’h’+"jedi"[1:2] + ’’.join((’l’*2).split())) + ’o’

Answer: ’hello’

(5) (10 != 200) and not True or not False

Answer: True

(6) 3 < 10 or 5

Answer: True

(7) (10 < 100) or (50 < 100/0)

Answer: True There is no divide-by-zero error because the first clause is True so the second isn’t even evaluated.

(8) len("hello") == 25/5 or 20/10

Answer: True

(9) -4*-2**3+-len(str(6))

Answer: 31 The ’-’ is negation here.

(10) 2*(2*(2*len("01")))

Answer: 16

(11) (len("expression")/10 + 15)/len(str(10+15))

Answer: 8.0

(12) (0*’x’) in ’Iowa’.upper()

Answer: True The empty string is always in any string.

(13) "w" in ’Iowa’ and (5!-4*3-7 or ’k’ not in "hawk")

Answer: Syntax error (!-)

(14) 15//2+1

Answer: 8

(15) (’foo’*2, (’a’<’b’) and True)

Answer: (’foofoo’, True)

(16) (not True, ’to’+’rn’, 72/6)[2-4]

Answer: ’torn’ The [2-4] is simply the index [-2].

(17) (7)[0:]

Answer: Syntax error (not a tuple) The singleton tuple would read (7,) instead.

(18) L[49//7%5].title()+L[0].upper()

Answer: ’OverTAKEN’

(19) { c:s.count(c) for c in ’’.join([w[1:] for w in s.split() if ’e’ not in w[-3:-1] and ’o’ in w[:4]]) }

Answer: {’a’: 2, ’e’: 7, ’d’: 1, ’k’: 2, ’m’: 2, ’o’: 3, ’n’: 1} This problem asks you to count letter occurrences in the original string but only for letters from a reduced subsequence of the string (’okeemonade’).

(20) ’+’.join(s.split()[2:3])[2:5] + s[32:37][1]

Answer: ’ai’ Many students were confused by the ’+’ which concatenates strings. This is different than the ’+’ used as "glue" in the join, although the latter turns out to be irrelevant given the slice selected.

(21) tuple({c for c in ’foobar’ if c != ’r’ } & { c for c in ’bingo’ if c != ’o’ })

Answer: (’b’,) These are sets, and ’&’ is set intersection.

(22) *# note: is there more than one answer? why or why not?*
’’.join(list({ c for c in ’foobar’ if c!= ’r’ } - { c for c in ’bingo’ if c != ’o’ }))

Answer: ’aof’ There may be many answers, because sets are unordered, thus permutations are allowed.

Find values for α and β that make each of the following expressions True. If no such values exist, explain why.

(1) ((1,4,3),[5,7,9],range(4,7,1))[α][β]%4

Answer: α = 1 β = 1 There are many answers here; some which are explicitly True (where the expression evaluates to 1: 0,0 1,0 1,2 2,1) and others which are defacto True (expression evaluates

to non-0 value other than 1: 0,2 1,1 2,2). The only ones that don't work are 0,1 and 2,0.

(2) $1+\alpha\%2 == (\alpha+2)\%2$

Answer: there is no solution; LHS will be 1 if even(a), 2 if odd(a); RHS will be 0 if even(a), 1 if odd(a)

(3) α in β and α != $\beta[\beta$.index(α)]

Answer: α **= 'foo'** β **= 'foobar'** The key issue here is that α needs to be a non-single-character substring found in β, so that the index expression returns the single starting character of α which will not be equal to α itself. Note that α could also be the null string '', which is always in any other string.

2. Python REPL (22 points)

Consider the following interative session with the Python REPL. What is output to the screen after each input?

```
>>> i = 9
>>> while i > 0:
        print(i)
        if (i%2 == 0):
            i = i/2
        i = i - 1
9
8
3.0
2.0
>>> n = 10
>>> while n <= 100:
        print(n)
        n = n + 2
        if n % 5 == 0:
            n = n - 1
            break
        print(n)
10
12
12
14
14
16
16
18
18
>>> def test(x=3, y=5, z=2):
        return(x-y+z)
>>> test()
0
```

```
>>> test(10)
7
>>> test(10,20)
-8
>>> test(20,10)
12
>>> test(z=35)
33
>>> test(10,z=35)
40
>>> def foobar(x, y=1):
        s = ''
        for i in range(0,x,y):
            s = s + str(i)
        print(s)
>>> x = foobar(6)
012345
>>> y = foobar(2)
01
>>> x + y
Error: NoneType + NoneType
```

3. Complete the Function (8 points)

(1) The *isslice*() function returns True if its first argument is a subsequence of its second argument, for any sequence type. So, for example:

```
>>> isslice('bam', "bimbamboo")
True
>>> isslice([21,22],range(10))
False
>>> isslice([0,0,0],[0,0,0])
True
```

Unfortunately, the definition is incomplete. Fill in the blanks to complete the definition.

```
def isslice(x, s):
    return(any([ x==s[i:i+len(x)] for i in range(len(s)) ]))
```

(2) The *invert*() function reverses its argument, which is a list, in place, that is, it makes structural changes to the list. So:

```
>>> L = list(range(4))
>>> L
[0, 1, 2, 3]
>>> invert(L)
>>> L
[3, 2, 1, 0]
```

Unfortunately, the definition is incomplete. Fill in the blanks to complete the definition.

```
def invert(L):
    for i in range(len(L)//2):
        L[i], L[len(L)-i-1] = L[len(L)-i-1], L[i]
```

An alternate solution simply indexes from the end.

```
def invert(L):
    for i in range(len(L)//2):
        L[i], L[-i-1] = L[-i-1], L[i]
```