CS1010E Final (AY2020/2021, SEM1)

Question #: 1

For question 1 to 10, evaluate the given Python expressions. What do you have if you type the give expression into the Python shell as the first command after started,

Evaluate: `10+6//4+3//2`

A. `13`
B. `12`
C. `13.0`
D. `3`
E. `3.5`

Question #: 2

Evaluate: `([1,2,3]*2)[::2]`

A. `[1,2,3]`
B. `[1,3,1,3]`
C. `[]`
D. Error
E. `[1, 3, 2]`

Question #: 3

Evaluate: `'abcdef'[(1,'d',4)[2]]`

A. `'c'`
B. `'f'`
C. `''`(empty string)
D. `'e'`
E. Error

Question #: 4

Evaluate: `[a,b,c][int('1')]`

A. `'a'`
B. Error
C. `'b'`
D. `'c'`
E. `''`(Empty string)

Question #: 5

Evaluate: `{'a':(1,2,3),'b':'qsr',2:[6]}['abc'[1]][0]`

A. `1`
B. Error
C. `6`
D. `'s'`
E. `'q'`

Question #: 6

Evaluate: `[i**2 for i in range(1,10)][3]`

A. `3`
B. `16`
C. `9`
D. `4`
E. `2`

Question #: 7

Evaluate: `len([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])`

A. `3`
B. `12`
C. `4`
D. `9`
E. Error

Question #: 8

Evaluate: `tuple(map(lambda x:x[0]+x[1],((1,2),(3,4))))`

A. `(1,2,3,4)`
B. `(3,7)`
C. `10`
D. `(4,6)`
E. `((1,2,3,4),)`

Question #: 9

Evaluate: `list(tuple(set('abcabc')))`

A. `['abcabc']`
B. `['b', 'c', 'a']`
C. `['abc’]`
D. `[(‘abc’,)]`
E. `[(‘abcabc’,)]`

Question #: 10

Evaluate: `(lambda x,y,z:x(y)+x(z))((lambda x:x*2),4,5)`

A. Error
B. `18`
C. `13`
D. `(9,18)`
E. `(8,10)`

Question #: 11

Questions 11 to 23 are about Output Prediction: If we put each of the given statements into a .py file and run. What will be the output?

What is the output of this following code?

```
d = {1:2,2:4,4:3,3:5,5:1}
x = 1
for i in range(11):
    x = d[x]
print(x)
```

A. Error
B. 1
C. 2
D. 3
E. 4

Question #: 12

What is the output of this following code?

```
def foo(x):

    while type(x) == list:

        x = x[0]

    return x

print(foo([[[[1]]]]))
```

A. Error
B. ([([1])])
C. 1
D. ([1])
E. ((1,),)

Question #: 13

What is the output of this following code?

```
a,b = 1,2

a,b = b,a

a,b = b,a

print(a)
```

A. Error
B. `2`
C. `1`
D. `(1,2)`
E. `(2,1)`

Question #: 14

What is the output of this following code?

```
def foo(a,b):

    if len(b) == 1:

        return b[0]

    return a(b[0],foo(a,b[1:]))

print(foo(lambda x,y:x+y,[i for i in range(1,5)]))
```

A. `[1,2,3,4]`
B. `5`
C. `10`
D. `4`
E. `[1,2,3,4,5]`

Question #: 15

What is the output of this following code?

```
ans = 0

for i in range(0,100,50):

    ans += i

print(ans)
```

A. `0`
B. `4950`
C. `50`
D. `100`
E. `150`

Question #: 16

What is the output of this following code?

```
l = [0,0,0]

k = [l]*2

k[1][0] = 1

print(sum(k[0]))
```

A. `0`
B. `2`
C. `1`
D. `(1,1)`
E. Error

Question #: 17

What is the output of this following code?

```
lf = [lambda x:x+i for i in range(0,10,2)]

print(lf[3](7))
```

A. Error!
B. `10`
C. `15`
D. `7`
E. `[7]`

Question #: 18

What is the output of this following code?

```
m = map(lambda x:x//3,[i for i in range(20)])

print(m[2])
```

A. Error
B. 0
C. `1`
D. `3`
E. `2`

Question #: 19

What is the output of this following code?

```
a,b = 1,2

a,b = b,a

a,b = b,a

print(a)
```

A. Error
B. `2`
C. `1`
D. `(1,2)`
E. `(2,1)`

Question #: 20

What is the output of this following code?

```
def foo(s):

    return sum(map(lambda x:1+ord(x.lower())-ord('a'),s))

print(foo('Cab'))
```

A. Error
B. `0`
C. `6`
D. `3`
E. `'cab'`

Question #: 21

What is the output of this following code?

```
class A:

    def foo():

        print('A')

class B(A):

    def foo():

        print('B')

class C(A):

    def foo():

        print('C')

class D(C):

    def foo():

        print('E')

class E(B):

    pass


E.foo()
```

A. Error!
B. A
C. B
D. C
E. (Print nothing)

Question #: 22

What will be the length of the list lst that will crash this function reverseList()?

```
def reverseList(lst):

     l2 = lst[:]

     return reverseList(lst[1:])+[l2[-1]]
```

A. Only when **`len(lst) == 0`**
B. Only when **`len(lst) == 1`**
C. Only when **`len(lst)`** is even
D. The code **will NOT crash** at all for any length of **`lst`**
E. The code **will crash** for any length of **`lst`**

Question #: 23

The following function tries to reverse some collections:

```
def reverse(a):
    if not a:
        return a
    return reverse(a[1:])+a[0]
```

Which of the following lines will NOT crash? (You can choose more than one answer.)

A. `reverse((1,2,3,4))`
B. `reverse([1,2,3,4])`
C. `reverse('abcde')`
D. `reverse({1,2,3,4})`
E. Every answer here will crash

Question #: 24

Questions 24 and 25 are about debugging.

We write a function **compareTwoList()** and expect it to compare if two lists contain the same integers, but the elements not necessarily in the same orders in the two lists. We assume that **l1** and **l2** contain integers only. Here is the function:

```
def compareTwoList(l1,l2):
    l1c = l1.copy()
    l2c = l2.copy()
    for i in l1:
        l1c.remove(i)
        if i in l2c:
            l2c.remove(i)
    return not l2c
```

What is true about the above code? (You can choose more than one answer.)

A. The function tells the right answer sometime. And sometimes it tells the wrong answer.
B. The function will always tell the wrong answer
C. The function will always tell the right answer
D. The function will not crash even if the inputs are empty lists.
E. The function will always crash.

Question #: 25

Assuming the input **l** is a list of integers. What is true about the following function? (May choose more than one answers.)

```
def foo(l):
    l2 = l[1:]+[min(l)]
    l3 = [(l[i],l2[i]) for i in range(len(l))]
    return not tuple(filter(lambda x:x[0]<x[1],l3))
```

A. It will return a sorted list in ascending order

✓B. It will crash if the input is an empty list

C. It will crash if the length of the input is 1

D. It will return `True` if the list is in non-descending order and `False` otherwise

✓E. It will return `True` if the list is in non-ascending order and `False` otherwise

Question #: 26

Given a list of integers, the following code is supposed to check if the whole list is greater than or equal to 0.

```
def checkAllPositive(L):
   for i in ___1___ :
      if i <___2___  :
         return ___3___
   return ___4__
```

Question #: 27

We want to implement our own version of the function filter. And we simplify the problem that we only need to work on lists and return lists. Fill in the blanks for correct answers.

```
def filter(f,lst):
   output = [ ]
   for ___1___ in ___2___  :
      if ___3___  :
         output.append(item)
   return output
```

Question #: 28

Given a sequence of integers, the following function is supposed to return a tuple that is the largest and the second largest of the sequence. E.g. findLargest2((2,1,3,5,6,5)) will return (6,5). Fill in the blanks for the missing variable in the following code. Please node that the answers are case sensitive and you should not have any space in your answer.

```
def findLargest2(L):
   first = max(L[0],L[1])
   second = min(L[0],L[1])
   largest = max(L)
   smallest = min(L)
   for i in L[2:] :
      if i >    1     :
         if i >    2     :
            ___3___   =  first
            ___4___   =  i
         else:
               ___5___  =  i
   return (first,second)
```

Question #: 29

Given a shallow list L (a list that does not contains any sequence.) Complete the following definition of function deepen(L) that will add more "layers" of list as follows:

```
>>> print(deepen([1,2,3,4,5]))
[1, [2, [3, [4, [5]]]]]
>>> print(deepen([]))
[]
>>> print(deepen([1]))
[1]
```

```
def deepen(L):
    if not L:
        return ___1___
    if len(L) == ___2___ :
        return ___3___
    return [ ___4___ , ___5___ ]
```

Question #: 30

Question OOP (10 marks)

Remember our RPG assignment? We are going to build more characters on top of that in this question.

Your code should demonstrate good OOP practices in this question.

Write the class for a new type of character **`TankFighter`**. He will have the same parameters (`str`, `cost` and so on) as a **`Fighter`**, except that the class name is '`TankFighter`' and his `str` is only 50. He will act as a Fighter does. However, when he was hurt, his damage is halved (round down to the smaller integer). For example, if another character, e.g. a Fighter, 'acts'(attacks) on a `TankFighter` with 101 damage. The hit point of the `TankFighter` will be reduced by 50 only.

The code of `Fighter` and `Character` are provided for your reference only.

```
class Character(object):
    def __init__(self):
        self.name = ''
        self.maxhp = 1000
        self.hp = 1000
        self.str = 0
        self.maxmana = 0
        self.mana = 0
        self.cost = 100
        self.alive = True
    def act(self,myTeam,enermy):
        return
    def gotHurt(self,damage):
        if damage >= self.hp:
            self.hp = 0
            self.alive = False
        else:
            self.hp -= damage

class Fighter(Character):
    def __init__(self):
        super().__init__()
        self.name = 'Fighter'
        self.str = 100

    def act(self,myTeam,enermy):
        target = randAlive(enermy)
        enermy[target].gotHurt(self.str)
```

Question #: 31

Question Find the Equivalent Resister (10 marks)
There is a part of a circuit that require an electric resistance of `o` ohms. However, there are two slots that you can plug in two resistors in serial. You have some resistors in your hand that you know the resistance values and their values are stored in a list `rl`. For example, if you got 4 resistors in your hand and their values are 1,3,5,7 ohms, and you need a total of 6 ohms, you can pick the two resistors with 1 and 5 ohms. Write a function `check(rl,o)` that takes in a list of the resistors values and the required ohms `o`, and return `True` if there are at least a pair of resistors that can provide the combined resistance of `o` ohms, or `False` otherwise. Here is an example output:

```
>>> print(check([1,3,5,7,9,11,13,15],24))
True
>>> print(check([1,3,5,7],5))
False
>>> print(check([1,3,5,7],6))
True
>>> print(check([12,13,15],24))
False
>>> print(check([12,13,12],24))
True
```

Answers

Qn	Ans	Qn	Ans	Qn	Ans
1	B	11	C	21	C
2	E	12	C	22	E
3	D	13	C	23	C
4	B	14	C	24	D
5	E	15	C	25	B, E
6	B	16	C	26	`L,0,False,True`
7	C	17	C	27	`item, lst, f(item)`
8	B	18	A	28	`second, first,` `second, first, second`
9	B	19	C	29	`L, 1, [L[0]],` `deepen(L[1:])`
10	B	20	C	30, 31	SKIPPED