CS1010E Final (AY2022/2023, SEM1)

Section 1 Syntax and Python usage

QN	Questions	Answer
	You can consider the code in each question is in a separate file. What will be the output when we run the file in IDLE?	
1	`print(9-2*3-1)`	a) 2 b) 20 c) 5 d) 4 e) 14
2	`print(3**--3)`	a) 27 b) 9 c) -9 d) 0 e) Error
3	`print(False or True)`	a) True b) False c) 1 d) 0 e) None of the rest
4	`print((False == False) in [False])`	a) False b) True c) 0 d) 1 e) Error
5	`print(True != False in [False])`	a) True b) False c) 1 d) 0 e) Error
6	`print(10>10 and 4<(9/0))`	a) False b) True c) 1 d) 0 e) Error
7	`print('abc'>'abbbbbbbb')`	a) True b) False c) 'abc' d) 'abcabbbbbbbb' e) Error
8	`print(max('12','111')+max('3','0'))`	a) 123 b) 111 c) 15 d) 114 e) 1113
9	`print('1234567890'[1])`	a) 2 b) 1 c) 0 d) 3 e) None
10	`print('12345'[1:5][0:4][2][0])`	a) 4 b) 1

		c) 2 d) 3 e) Error
11	print((2)+(3,))	a) Error b) (2, 3) c) (5,) d) 5 e) (2, (3,))
12	print([9,[8,[7,6],[5,4]],3][1][2][3:5])	a) [] b) 5 c) 4 d) 6 e) Error
13	print(['123','456']['012'[1]])	a) Error b) Empty string c) 456 d) 123 e) 4
14	A = ['a','b'] B = [A,'c'] C = [B,A] B[0].remove('a') print(C)	a) [[['b'], 'c'], ['b']] b) [['b', 'c'], ['b']] c) [[['b'], 'c'], ['a','b']] d) [[['a','b'], 'c'], ['b']] e) [['b', 'c'], ['a','b']]
15	L1 = [0,1] L2 = [L1,2,3] L3 = L2.copy() L3[0][1] = 999 print(L2)	a) [[0, 999], 2, 3] b) [[0, 1], 2, 3] c) [[999, 1], 2, 3] d) [999, 2, 3] e) Error
16	L1 = ([1,2],3,4) def f(L): L[0][0] = 9 f(L1) print(L1)	a) ([9, 2], 3, 4) b) ([1, 2], 3, 4) c) [[1, 2], 3, 4] d) (9, 3, 4) e) Error
17	B = [1,2] B.append(B) print(B[2][2][2][2][2][2][0])	a) 1 b) [1, 2] c) [1, 2, [...]] d) Error e) [[1, 2, [...]]]
18	print((9,8,7)*2+(1,2,3)[1])	a) Error b) (9, 8, 7, 1) c) (9, 8, 7, 2) d) (11, 10 , 9) e) (9, 8, 7, 9, 8, 7, 2)
19	A = ([1,2,3],[5,6,7]) A[0].append(4) print(A)	a) ([1, 2, 3, 4], [5, 6, 7]) b) ([1, 2, 3], [5, 6, 7]) c) Error d) ([1, 2, 3], [5, 6, 7], 4) e) [[1, 2, 3, 4], [5, 6, 7]]
20	def f(x): return x+2 print(f(f(f(f(7)))))	a) 15 b) 13 c) 7 d) 28 e) 14

21	def a(x):
	    return 1+c(x)
	def b(x):
		a) RecursionError
		b) NameError
		c) Infinite loop

	`    return 2+d(x)`
	`def c(x):`
	`    return 1+a(x)`
	`print(a(1))`
		d) 8
		e) 6

22	`def f(x):`
	`    if x > 0:`
	`        return 1+f(x-2)`
	`print(f(10))`
		a) TypeError
		b) None
		c) 0
		d) 5
		e) RecursionError

23	`x = 100`
	`for j in range(0,100,10):`
	`    x -=1`
	`print(x)`
		a) 90
		b) 0
		c) 91
		d) 1
		e) 100

24	`x = 0`
	`for i in range(0,12):`
	`    for j in range(0,10):`
	`        for k in range(0,3):`
	`            x += 1`
	`print(x)`
		a) 360
		b) 0
		c) 25
		d) 198
		e) 22

25	`x,y = 0,0`
	`while x < 10:`
	`    x += 2`
	`    y += 3`
	`print(y)`
		a) 15
		b) 10
		c) 12
		d) 8
		e) 27

26	`x = 0`
	`for i in range(0,1000000,25):`
	`   x = 1-x`
	`print(x)`
		a) 0
		b) RecursionError
		c) 40000
		d) - 40000
		e) 1

27	`x = 10`
	`if x > 10:`
	`  y = 1`
	`elif x < 100:`
	`  y = 2`
	`elif x == 10:`
	`  y = 3`
	`else:`
	`  y = 4`
	`print(y)`
		a) 2
		b) 1
		c) 3
		d) 4
		e) Error

28	`y = 'a'`
	`if y == 'b':`
	`    x = 1`
	`elif y == 'a':`
	`    x = 2`
	`if y == 'c':`
	`    x = 3`
	`else:`
	`    x = 4`
	`print(x)`
		a) 4
		b) 2
		c) 3
		d) 1
		e) Error

29	`print(tuple({9:8,7:6,5:4}))`	a) (9, 7, 5) b) (9, 8, 7, 6, 5, 4) c) ((9, 8), (7, 6), (5, 4)) d) ({9:8,7:6,5:4},) e) ({9:8,7:6,5:4})

30

```
d = {0:2, 1:5, 2:1, 3:2, 4:1, 5:3, 3:9,
2:7, 2:3}
a, output = 0, ''
while a in d:
    a = d[a]
    output += str(a)
print(output)
```

a) 239
b) 27
c) 21539
d) Infinite Loop
e) 20

31

```
f1 = lambda x,y : x+y
f2 = lambda a,b,c: a(b,c)
print(f2(f1,4,7))
```

a) 11
b) 28
c) a function
d) Error
e) None

32

```
f = lambda x:x%4
print(len(list(filter(f,range(0,10)))))
```

a) 7
b) 3
c) 10
d) 12
e) 1

33

```
print(list(map(lambda x:x*3,'123')))
```

a) ['111', '222', '333']
b) ['123123123']
c) '123123123'
d) ['123', '123', '123']
e) Error

34

```
class Graduate:
    def __init__(self,name,title):
        self.name = name
        self.title = title
        self.fullname = title+" "+name


alan = Graduate('Alan','BSc')
alan.title = 'Dr'
print(alan.fullname)
```

a) BSc Alan
b) Dr Alan
c) Alan
d) Dr
e) Bsc

35

```
class Animal:
    def __init__(self):
        self.sound = None
    def speak(self):
        print(self.sound)
class Dog(Animal):
    def __init__(self):
        self.sound = "Woof"
class Chihuahua(Animal):
    def speak(self):
        print("*cute*")
        super().speak()
dolly = Chihuahua()
dolly.speak()
```

a) *cute*
   None
b) *cute*
   Woof
c) Woof
d) *cute*
e) None

36

```
n = 2
x = 0
try:
    while True:
        x += 10//n
        n-=1
except:
    print(x)
```

a) 15
b) None
c) 10
d) 5
e) The code crashes

37

```
n = 0
try:
```

a) 3
b) 1

	n += 1 except what: n += 1 except: n += 1 else: n += 1 finally: n += 1 print(n)	c) 2 d) 4 e) The code crashes
38	x = 0 try: assert 0*(1/0) except AssertionError: x = 1 except ZeroDivisionError: x = 2 except: x = 3 print(x)	a) 2 b) 0 c) 1 d) 3 e) The code crashes
39	The two file opening modes, 'w+' and 'r+', have some different functionalities.	a) True b) False
40	print(round(2.5)+round(3.5))	a) 6 b) 5 c) 7 d) 7.0 e) 6.0

Section 2

	New section	

41	If the input L is a list of numbers with odd length, what is the output of the following code?
	`def foo(L):`
	`  L2 = list(L.copy())`
	`  output = []`
	`  while len(L2)>2:`
	`  output.append(max(L2))`
	`  output.append(min(L2))`
	`  L2.remove(max(L2))`
	`  L2.remove(min(L2))`
	`  output.append(L2[0])`
	`  return output[-1]`
		a) Median of L
		b) Mean of L
		c) Mode of L
		d) A sorted list of L
		e) A zig-zag list of L

42	If we want to write a Python program to store a large collection of names(strings) and query if a certain name(string) is in the collection, what will be the best data structure to store all the names in terms of speed?	a) Dictionary b) List c) Tuple d) File e) All of the answer here will be the same

43	Give a list of unique integers L, we want to find how many pairs of number in L with a sum to a value x. E.g.
	`>>> sumTo([1,2,3,4,5,6,7,8,9],13)`
	`3`
	Explain why the following code is buggy?
	`def sumTo(L,x):`
	`  count = 0`
	`  for i in range(len(L)):`
	`  for j in range(len(L)):`
	`  if L[i]+L[j]==x:`
	`  count += 1`
	`  return count`
		a) Each pair may be counted twice
		b) The code is actually correct all the time
		c) The code will crash if the length of the list is 0
		d) The code will be correct if x//2 is not in L
		e) The code will be correct if x//2 is in L

44	A palindrome is a word that reads the same backwards as forwards, e.g. madam, ere. The following code is to check if a string is a palindrome:
	`def check_palindrome(s):`
	`  if not s:`
	`  return True`
	`  if s[0]==s[-1]:`
	`  return check_palindrome(s[1:len(s)-1])`
	`  return False`
	Provided that the input s is a string, what is the best description of the code?
		a) The code can check palindrome correctly without crash
		b) The code actually tell the wrong (reversed) answer always
		c) The code will crash if len(s) is odd
		d) The code will crash if the input s has its length len(s) == 1800
		e) The code cannot work for reasons that are not mentioned here

Section 3 Fill-in-the-blanks

Please note that your code will got zero mark if there are syntax errors. Also, There is no partial marks for each question.

	For all the fill-in-the-blank questions, please do not enter spaces before (on the left side) of your answers. Or it will be deemed as wrong answer	

45	Given a list of unique numbers L, we want to check if the list L is sorted in an ascending order. E.g.
	`>>> print(check_ascending([1,2,3]))`
	`True`
	`>>> print(check_ascending([1,4,3]))`
	`False`
	Complete the following code for the function:
	`def check_ascending(L):`
	`  for i in (__BLANK1__):`
	`  if (__BLANK2__):`
	`  return (__BLANK3__)`
	`  return (__BLANK4__)`	

46	Implement a sorting function to sort a list of numbers by filling in the blanks. Your function should modify the input list and sort it without returning a new list. Sample output:
	`>>> L = [4,1,2,3,8,1,2,3,9,6]`
	`>>> my_sort(L)`
	`>>> print(L)`
	`[1, 1, 2, 2, 3, 3, 4, 6, 8, 9]`
	Fill in the blanks the blanks:
	`def my_sort(L):`
	`  for i in (__BLANK1__):`
	`  for j in (__BLANK2__):`
	`  if (__BLANK3__):`
	`  L[BLANK4],L[BLANK5] = L[BLANK6],L[BLANK7]`
	Hint: You should use selection sort or bubble sort	

47	You are given a 2D map like the following:
	`map1 = ['.#+#...#...',`
	`  '..#...#+#..',`
	`  '.......#...',`
	`  '.+...#.....',`
	`  '....#+#...#',`
	`  '.....#...#+']`
	And you want to locate all the positions of the star by printing out the positions of their centers :
	`  #`
	`#+#`
	`  #`
	E.g.
	`>>> find_star(map1)`
	`(1, 7)`
	`(4, 5)`	

	Fill in the blanks in the following code to complete the function `find_star()`:
	`def find_star(m):`
	`  n_row = len(m)`
	`  n_col = len(m[0])`
	`  for i in (__1__):`
	`  for j in (__2__):`
	`  if (__3__):`
	`  print((i,j))`	

48	The following is the very slow code for our recursive binomial coefficient computation in Assignment 3.
	`def nCk(n,k):`
	`  if n==k or k==0:`
	`  return 1`
	`  ans = nCk(n-1,k-1)+nCk(n-1,k)`
	`  return ans`
	Use one technique taught in our class to speed up the code by filling in the blanks:
	`X = __BLANK1__`
	`def nCk(n,k):`
	`  if (__BLANK2__):`
	`  return (__BLANK3__)`
	`  if n==k or k==0:`
	`  return 1`
	`  ans = nCk(n-1,k-1)+nCk(n-1,k)`
	`  X[__BLANK4__]=__BLANK5__`
	`  return ans`	