NATIONAL UNIVERSITY OF SINGAPORE

MIDTERM ASSESSMENT FOR
CS1010E – PROGRAMMING METHODOLOGY

(Semester 1: AY2019/2020)

Time allowed: 1 hour

INSTRUCTIONS TO CANDIDATES

1. This assessment paper contains **TWENTY FIVE(25)** multiple-choice questions (MCQ) and comprises of **TWELVE (12)** printed pages, including this page.
2. All questions carry **4 marks** each. The maximum possible score is **100 marks**.
3. This is a **CLOSED BOOK (WITH AUTHORIZED MATERIALS)** assessment. You may bring in one piece of A4 size reference sheet.
4. Calculators are allowed, but not laptops or other electronic devices.
5. Answer all **MCQ questions** by shading the letter corresponding to the most appropriate answer on the **OCR form** provided. Shade and write down your student number on the **OCR form** as well. Use a 2B pencil to shade on the OCR form, or the grading machine might not be able to register your shading.
6. Submit the **OCR form** at the end of the assessment. You may keep the question paper.
7. Leave your student card on the desk throughout this assessment.
8. Do **NOT** look at the questions until you are told to do so.

1. [arithmetic] Suppose x, y and z are three positive integers and $x > y > z$. What is the maximum possible value assigned to variable i after the following statement is executed?

```
i = x % y % z
```

A. $x - 1$
B. z
C. $z - 1$
D. $y - 1$
E. None of the above

2. [arithmetic] Given that **x** can be any integer value, which of the following expressions will always evaluate to the same value as evaluating the expression **2 < x < 6** ?

A. `(2 < x) < 6`
B. `2 < x or x < 6`
C. `2 < x and x < 6`
D. `2 < (x < 6)`
E. There are more than one correct answer

3. [simple logic] What is printed by the following code fragment?

```
i = 1
while i < 10:
    if (i%5 == 0):
        break
    i += 1
print(i, ' == 10')
```

A. `True`
B. `10 == 10`
C. `5 == 10`
D. `False`
E. None of the above

4. [Boolean logic] What does the following function **mix** return?

```
def mix(x,y,z):
    if x <= y:
        if z <= x:
            return z
        else:
            return x
    elif y <= z:
        return y
    return x if x <= z else z
```

A. It returns the smallest value among the values stored in variables **x**, **y** and **z**.
B. It returns the middle value among the values stored in variables **x**, **y** and **z**.
C. It returns the largest value among the values stored in variables **x**, **y** and **z**.
D. It contains syntax error that makes the code not executable.
E. None of the above

5. [short-circuiting] What is TRUE about the following code fragment?

```
x = 20
n = int(input())
if n != 0 and x // n > 6:
    x += 1
else:
    x %= n
print(x)
```

A. A syntax error will be reported when the code is loaded into the shell.
B. A runtime error will occur if user input is **0**.
C. The program will print out **0** if user input is **5**.
D. The program will print out **21** if user input is **0**.
E. None of the above.

6. [short circuiting] What is TRUE about the following code fragment?

```
x = 5
n = int(input())
if n < 0 or not(n%2) or x // n < 6:
    x += 1
else:
    x //= 2
print(x)
```

A. A syntax error will be reported when the code is loaded into the shell.
B. A runtime error will occur when some specific integer value is input to **n**.
C. The fourth line (x+=1) will be executed always for any integer value input to **n**
D. The code will print out **10** if user input is **3**.
E. None of the above.

7. [arithmetic] What is the returned value if the following function is called with argument 50?

```
def count4(n):
    cnt = [0] * 4
    for i in range(n+1):
        if i%2 == 0:
            if i%6 == 0:
                cnt[0] += 1
            else:
                cnt[1] += 1
        elif i%3 == 0:
            cnt[2] += 1
        else:
            cnt[3] += 1
    return cnt
```

A. `[8,17,8,18]`
B. `[9,17,9,16]`
C. `[8,17,8,17]`
D. `[9,17,8,17]`
E. None of the above

8. [number manipulation] Function **mys()** has the following input-output relationship:

num	2222	3537	2124	658
mys(num)	0	1111	100	10

```
def mys(num):
    bin = 0
    k = 1
    while (num > 0):
        d = num % 10 % 2
        num //= 10
        bin = XXXXXXXXX
        k *= 10
    return bin
```

Which of the following expressions can be substituted into **XXXXXXXXX** to produce the desired output?

A. `bin * k + d`

B. `bin * k + (1-d)`

C. `bin + (0 if d else d) * k`

D. `bin + d%2 * k`

E. None of the above

9. [iteration] Which one of the following statements is TRUE about the following code fragment?

```
x = 1
y = 1
while (x < 300):
   x *= 3
   y += 1
```

A. The value of **x** after the iterations is 300.

B. The value of **y** after the iterations is 6.

C. The while statement goes into infinite loop.

D. There is syntax error in the statement: **x *= 3**.

E. None of the above.

10. [iteration] What is printed by the following code fragment?

```
sum = 0
for i in range(0,10):
    sum = sum + i
    i = i + 5
print(sum)
```

A. 0

B. 45

C. 5

D. Run-time error

E. None of the above

11. [number manipulation] The following function definition aims to determine the pair of adjacent digits in a number **num** that forms biggest number among all the pairs form. Its behavior is described in the following executions:

```
>>> maxPair(1038491390)
91
>>> maxPair(103849190)
91
```

```
def maxPair(num): # assume num >= 10
    maxd = 0
    while num > 10:
        pair = num % 100
        if pair > maxd:
            maxd = pair
            <missing statement>
    return maxd
```

Which of the following statements should be used to fill in the <missing statement> to complete the definition of **maxPair**?

A. `return num // 100`
B. `num = num / 100`
C. `num = num // 10`
D. `num = num // 100`
E. None of the above

12. [iteration, arithmetic] What is the value returned from calling the following function as such: `tupling(3,10)`?

```
def tupling(x,y):
    cnt = 0
    for i in range(1,x):
        for j in range(0,y,i):
            cnt += 1
    return cnt
```

A. 14
B. 15
C. 16
D. 18
E. None of the above

13. [recursion] Consider the following definition of recursive function **rec** .

```
def into(aStg, acc):
    if aStg:
        return into(aStg[1:],acc+1)+str(acc)
    else:
        return str(acc)
```

Which of the following strings will be returned from the execution of the function call **rec('abcd',0)**?

A. '01234'

B. '500000'

C. '54321'

D. '543210'

E. None of the above.

14. [list, iteration] In order to have the function **more** behaves as shown below:

```
>>> more([1,2,3,4,5])
[1, 3, 6, 10, 15]
>>> more([1,1,1,1])
[1, 2, 3, 4]
>>>
```

Which of the following statements should be used to fill in the missing statement in the following definition of function **more**?

```
def more(lst):
    newlst = []
    tmp = 0
    for num in lst:
        tmp = tmp + num
         <missing statement>
    return newlst
```

A. newlst + [tmp]

B. newlst.extend(tmp)

C. newlst.append(tmp)

D. All of the above

E. None of the above

15. [list, algorithm] Given a list containing just zeros and ones, the following (currently not fully defined) function re-orders all elements in a list such that all zeros are at the left side of ones. Thus, we have the shell interaction:

```
>>> list1 = [0,1,0,1,0,0,1,0]
>>> ordered(list1)
>>> list1
[0, 0, 0, 0, 0, 1, 1, 1]
>>>
```

```
def ordered(lst):
    j = len(lst)-1
    i = 0
    while (i < j):
        if lst[i] == 0:
            i += 1
        elif lst[j] == 1:
            j -= 1
        else:
            lst[i],lst[j] = lst[j], lst[i]

            < missing statement >
```

In the following list of statements, determine ALL the statements that can be used individually to fill in the missing statement to complete the function definition.

(i)	`i += 1`	(ii)	`j -= 1`
(iii)	`break`	(iv)	`continue`
(v)	`i,j = i+1,j-1`	(vi)	`i,j = i-1,j+1`

A. Statement (v) only
B. Statement (vi) only
C. Statements (i), (ii), (v) only
D. Statements (i) and (ii) only
E. None of the above.

16. [dictionary] Given `dd = { 1:{2:3}, 2:{3:1}, 3:{1:3} }`, what is the result of evaluating `max(dd[dd[2][3]])`?

A. `3`
B. `2`
C. `{2 : 3}`
D. Error
E. None of the above

17. [slicing] Consider the following definition:

```
s1 = '1234567890'
```

The following string slicing expressions all attempt to produce the string `'975'`. How many of them actually produce the desired string?

```
(a) s1[8:2:-12]   (c) s1[-2:2:-2]   (e) s1[8:-7:-2]
(b) s1[8:3:-2]    (d) s1[-2:3:-2]   (f) s1[8:4:-2]
```

A. 2
B. 3
C. 4
D. 5
E. None of the above

18. [iteration, sequence] The following incomplete function **`morethan`** takes in a sequence of numbers **`seq`**, and a number **`diff`**, and determines if the difference in every adjacent pairs of the numbers is always more than **`diff`**. Its behavior is described in the following executions:

```
>>> morethan([13,74,21,64,90],23)
True
>>> morethan([21,17,34,26,0],20)
False
>>>
```

```
def morethan(seq, diff): #assume len(seq) >= 2
    for i in <exp>:
        if abs(seq[i] - seq[i-1]) > diff:
            continue
        else:
            return False
    return True
```

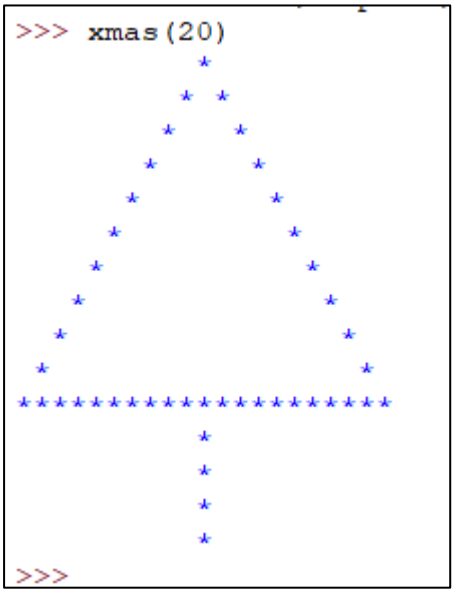
What expression is to be filled in **<exp>** to complete the definition of **`morethan`**?

A. `range(len(seq))`
B. `range(1,len(seq))`
C. `seq`
D. `seq[1:]`
E. None of the above

19. [algorithm] The following incomplete function definition draws a Christmas tree.

```
def xmas(width):
    height = 1
    stars = '*'
    left  = ' '* (width//2)
    right = ' '* (width//2)
    while len(left) > 0:
        print(left + stars + right)
        left = left[1:]
        right = right[1:]
        <missing statement>
        height += 1
    print('*'*(2*height-1))
    while height > 0:
        print(' '* (width//2) + '*')
        height = height // 2
```

Evaluating xmas(20) should produce the following picture:



Which of the following statements should be used to replace the `<missing statement>` in order to produce the picture?

A. `stars = '*' + ' '* height + '*'`
B. `stars = '*' + ' '*(2*height+1) + '*'`
C. `stars = '*' + ' '*(2*height) + '*'`
D. `stars = '*' + ' '*(2*height-1) + '*'`
E. None of the above

20. [recursion] Consider the following definition of recursive function **rec** .

```
def rec(stg, acc):
    if stg:
        return rec(stg[1:],acc+'a')+'b'
    else:
        return acc
```

Which of the following strings will be returned from the execution of the function call **rec('1234','')**?

A. **'aaaabbbb'**

B. **'abababab'**

C. **'aaaab'**

D. **'abbbb'**

E. None of the above.

21. [recursion, HOF] Consider the following function definition, what is the result of execution the function call **f(5, lambda x: x+2)**?

```
def f(n,g):
    if n > 1:
        return f(n-1, lambda x: g(n))
    else:
        return g(n)
```

A. **1**

B. **3**

C. **5**

D. **7**

E. Run time error occurs

22. [HOF] Consider the following function definitions, what is the result of execution the function call **h(3)** ?

```
def circle(f,g):
    return lambda x : g (f (x) )

g = lambda x : (x,x)

h = circle(g,g)
```

A. **((3,3),)**

B. **(3,(3,3))**

C. **((3,3),3)**

D. **((3,3),(3,3))**

E. None of the above

23. [series, recursion] Consider the following summation:

$$sum1(n) = \sum_{i=1}^{n} 2(i-1)$$

The following recursive function definition of **sum1** aims to compute the summation above. Complete the definition of **sum1** by choosing one of the statements provided.

```
def sum1(n):
    if n == 1:
        return 0
    else:
        <missing statement>
```

A. `return sum1(n-1)`
B. `return sum1(n-1) + 2*(i-1)`
C. `return sum1(n-1) + 2*(n-1)`
D. `return 2*(n-1)`
E. None of the above

24. [dictionary] Following is the definition of dictionary **dict** and a tuple of three expressions that manipulate **dict**,

```
dict = { 1 : 'Z', 2 : 'MAX', 3 : 'IS',
         4 : 'DEF', 5 : 'UN', 6 : 'A' }
(dict[max(dict)] == min(dict.values()),
 dict[3] == dict.pop(3),
 dict.pop(min(dict.keys())) == max(dict.values()))
```

Which of the following is the correct values returned from evaluating the tuple?

A. `(False,False,False)`
B. `(True, False,False)`
C. `(True, True, False)`
D. `(True, True, True)`
E. Runtime error arises

25. [dictionary, list comprehension] Given the following definition and an assignment to variable **dd**, what is the result of evaluating the call **`valKeys(dd,[0,2])`** (the ordering of the elements in the resulting list is not important)?

```
dd = {'a': 0, 'b': 1, 'c': 2,
      'd': 0, 'e': 1, 'f': 2}
def valKeys(dict,valLst):
    kys = []
    for val in valLst:
        kys.extend([k for (k,val) in dict.items()])
    return kys
```

A. `None`

B. `['a', 'b', 'c', 'd', 'e', 'f']`

C. `['a', 'b', 'c', 'd', 'e', 'f', 'a', 'b', 'c', 'd', 'e', 'f']`

D. `['a', 'd', 'c', 'f']`

E. None of the above

=== END OF PAPER ===