Introduction to Programming Sections A and B Mid-semester exam, Feb 20, 2021

Instructions:

- Total 90 marks, total time allowed 90 min.
- 15 MCQ questions, each 3 marks – these carry negative mark -1 for each wrong answer.
- Plus 5 questions, each 9 marks, that requires you to do more.
- Where necessary, assume version of Python to be version 3.x.
- Strict action will be taken against you if you are found to use unfair means.
- You may use a blank sheet to do rough work, and a calculator, but nothing else.

Part 1: Questions that require you to write a Python code or pseudo-code or add lines of code in a suggested Python program. Each question carries 8 marks.

A. Write a Python program to input N integers, and compute the sum of all even numbers. For example if N = 5, and the integers that were input were: 45, 75, 96, 34, 7. Then the result should be 96 + 34, or 130. An outline of the main program is given below.

```
#    Python program to input N (number of integers), and to input N integers.
#    While inputting the integers compute sum of all even numbers.
#    Then print the resulting sum.
#    You may test your program with N = 5, and integers: 45, 75, 96, 34, 7.
#    Replace this line with one or more lines of your code.
```

The correct code is:

```
#    Python program to input N (number of integers), and to input N integers.
#    While inputting the integers compute sum of all even numbers.
#    Then print the resulting sum.
#    You may test your program with N = 5, and integers: 45, 75, 96, 34, 7.
N = int(input("Input number of integers "))
sum = 0
k = 1
while k <= N:
    x = int(input("Input an integer "))
    if x%2 == 0:
        sum = sum + x
    k = k + 1
print(sum)
```

Case	Expected Output
5 45 75 96 34 7	130
1 1	0
1 2	2
4 1 2 2 -2	2
4	0

1 3 5 7	
0	0
5 0 -2 -4 -6 -7	-12

B. Complete the following function, which should work as follows. Given a list of integers, compute the sum of only those integers that are divisible by either 3 or 5. Use recursion and do not use loops. computeSum([1, 2, 3, 5]) would give 8 and computeSum([2, 0, 15, -3]) would give 12.

```
def computeSum(lst):
#   Replace this line with one or more lines of your code.
```

The correct code is:

```
def computeSum(lst):
    if len(lst) == 0:
        return 0
    if len(lst) == 1:
        return isDivisibleBy3_5(lst[0])
    else:
        return isDivisibleBy3_5(lst[0]) + computeSum(lst[1:])

def isDivisibleBy3_5(n):
    if n%3 == 0 or n%5 == 0:
        return n
    else:
        return 0

lst = [1,2,3,4,5]
print(computeSum(lst))
```

Case	Expected Output
[1,2,3,4,5]	8
[3,3,3,5]	14
[0,0,1,5,5]	10
[10,2,1]	10
[-5,-15,-20]	-40
[]	0

C. Complete the Python program given below. This uses a recursive function **gcd(a, b)** to compute the GCD(a, b), where **a > 0, b > 0**. By definition

GCD(a, b) = b, if a%b == 0,

GCD(a, b) = GCD(b, a%b), otherwise.

```
# This program helps compute gcd(a, b). Assume a > 0, b > 0
def gcd(a, b):
 # replace this with one or more lines to complete function gcd
# test statements
print(gcd(15, 4))
print(gcd(21, 12))
```

The correct answer is:

```
# This program helps compute gcd(a, b). Assume a > 0, b > 0
def gcd(a, b):
    if a%b == 0:
        return(b)
    else:
        return(gcd(b, a%b))
# test statements
print(gcd(15, 4))
print(gcd(21, 12))
```

Case	Expected Output
gcd(1,1)	1
gcd(15,4)	1
gcd(21,12)	3
gcd(24826148, 45296490)	526
gcd(16,4)	4

D. Complete the following code using **for** loop to check whether the given string is a valid email based on the following conditions

a. It can contain any number, including zero, lower-case English letters
b. It should contain at least one upper-case English letter
c. It should contain at least two digits
d. It should contain exactly one "@" symbol and exactly one "." (dot)

Do not use any builtin functions. For example, the string "this is valid email09K@gmail.com" is a valid email where as, the string "this is invalid email@gmail.com" is invalid.

```
def isValidEmail(s):
# replace this with one or more lines to complete the function
```

The correct answer is:

```
def isValidEmail(s):
    countDigits = 0
    countRate = 0
    countCap = 0
    countDot = 0
    for c in s:
        if c in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
```

```
            countCap = countCap + 1
        elif c in "0123456789":
            countDigits = countDigits + 1
        elif c in "@":
            countRate = countRate + 1
        elif c in ".":
            countDot = countDot + 1
    if countCap >= 1 and countDigits >= 2 and countRate == 1 and countDot == 1:
        return True
    else:
        return False
```

```
s ="Ip2021@gmail.com"
print(isValidEmail(s))
```

Case	Expected Output
"Ip2021@gmail.com"	True
"introduction5@gmail.com"	False
"56.Programming@gmail@com"	False
"Python21.gmail@com"	True
"Oemail@Gmail.com"	False
"this is valid email09K@gmail.com"	True
""	False
"2020TEXT.2021@"	True

E. Write a function **`remDuplicates(L)`** to remove all duplicate objects (in this case strings) from the list of strings, **`L`**. As an example, let list **`L = ['alpha', 'beta', 'gamma', 'alpha', 'delta']`**. Then after calling **`remDuplicates(L)`** to remove duplicate words the list **`L = ['alpha', 'beta', 'gamma', 'delta']`**. You are also required to add statements in the main program to call with actual parameter **`L = ['alpha', 'beta', 'gamma', 'alpha', 'delta']`**.

An outline of the overall program (main and function **`remDuplicates(L)`** is given below.

The correct code is:

```
#  Python program to remove all duplicate strings from the list, L.
def remDuplicates(L):
    temp=[]
    for i in L:
        if (i not in temp):
            temp.append(i)
    return(temp)
# test statements
L = ['alpha', 'beta', 'gamma', 'alpha', 'delta']
print(L)
L = remDuplicates(L)
```

```
print(L)
```

Case	Expected Output
['alpha' , 'beta' , 'alpha' , 'gamma' , 'delta']	['alpha' , 'beta' , 'alpha' , 'gamma' , 'delta'] ['alpha' , 'beta' , 'gamma' , 'delta']
['IIITD' , 'IIITB' , 'IIITA' , 'IIITH']	['IIITD' , 'IIITB' , 'IIITA' , 'IIITH'] ['IIITD' , 'IIITB' , 'IIITA' , 'IIITH']
['1' , '1', '1' , '1']	['1' , '1', '1' , '1'] ['1']
[]	[]
[“”, “”, “”]	[“”]

Introduction to Programming Sections A and B Mid-semester exam, Feb 20, 2021 - Part 2

Part 2: Multiple-choice questions, each 3 marks.

1. Here is a piece of code in Python, dealing with strings. Which of the lines in the code are erroneous (viz. in error)?

```
x = "IP A"                          # line no. 1
y = "EASY CSE101-SecB"              # line no. 2
print(x[0:2] + y[12:-1])            # line no. 3
x[-1] = "B"                         # line no. 4
y[-2] = "C"                         # line no. 5
```

   - ○ Line 3 only
   - ○ Line 4 only
   - ○ Line 5 only
   - ○ Lines 3 and 4 only
   - ○ Lines 3 and 5 only
   - ○ Lines 4 and 5 only
   - ○ Lines 3, 4 and 5 only
   - ○ None of the above
   - ○ All of the above

2. Let list `L = [6, 23, 3, 2, 0, 9, 8, 75].` Then, which of the following statement or statements will produce an output `[23, 2, 9, 75]` ?

   - ○ `print(L[1:7:2])`
   - ○ `print(L[0:7:2])`
   - ○ `print(L[1:8:2])`
   - ○ `print(L[0:8:2])`
   - ○ None of the above
   - ○ All of the above

3. Here is a piece of code in Python, using a **`while`** loop, and a **`break`** statement. What is the output once the code is executed?

```
i = 5
while True:
    if i%11 == 0:
        break
    print(i)
    i = i + 1
```

   - ○ `5 6 7 8 9 10`
   - ○ `5 6 7 8 9 10 11`
   - ○ `1 2 3 4 5 6 7 8 9 10`
   - ○ `1 2 3 4 5 6 7 8 9 10 11`
   - ○ None of the above
   - ○ …**`error`**…

4. Here is a Python code to define a function **`ToH`** to solve “Tower of Hanoi” puzzle.

How many times is this function **ToH** called and executed once we call this function as **`ToH(3, 'A', 'B', 'C')`** from the main program as shown below? INCLUDE the first time ToH is executed when called from the main program. NOTE: we are **moving 3 disks** from peg **A** to peg **B**.

```
# Solution to Tower of Hanoi puzzle
def ToH(n, src, dest, aux):
    if n==1:
        print("Move disk 1 from peg", src, "to peg", dest)
        return
    ToH(n-1, src, aux, dest)
    print("Move disk", n, "from peg",src, "to peg", dest)
    ToH(n-1, aux, dest, src)
#
# statement to call ToH from main program
ToH(3, 'A', 'B', 'C')
```

- 5
- 6
- 7
- 8
- None of the above

5. What is the output of the following code?

```
def g():
    y = 12
    def f():
        x = 6
        global y
        print(x+y)
    global x
    print(x)
    f()
    x = "g"
    print(x)
#
x = 5
y = 10
g()
```

- 5 16 g
- 5 15 g
- 5 17 5
- g 18 g
- Error

Here option A and E , both are correct.

6. Here is a piece of code in Python that executes function **`test`** recursively. What is the output once the code is executed?

```
# Python code to test
```

```
def test(i, j):
    if(i == 0):
        return j
    else:
        return test(i - 1, j + 1)
# statement to call test from main program
print(test(4, 7))
```

- 4
- 7
- 9
- 11
- 17
- None of the above
- … **error** … goes into an infinite loop

7. Here is an incomplete piece of code in Python. Which of the statements given below is correct?

```
if (x > 2):
    x = x * 2
if (x > 4):
    x = 0
print(x)
```

- Once the code executes, resulting x will always be equal to 0 independent of the initial value of x
- Once the code executes, resulting x will be double the initial value of x, provided x > 2 initially
- Once the code executes, resulting x will be 0, provided x > 2 initially
- None of the above

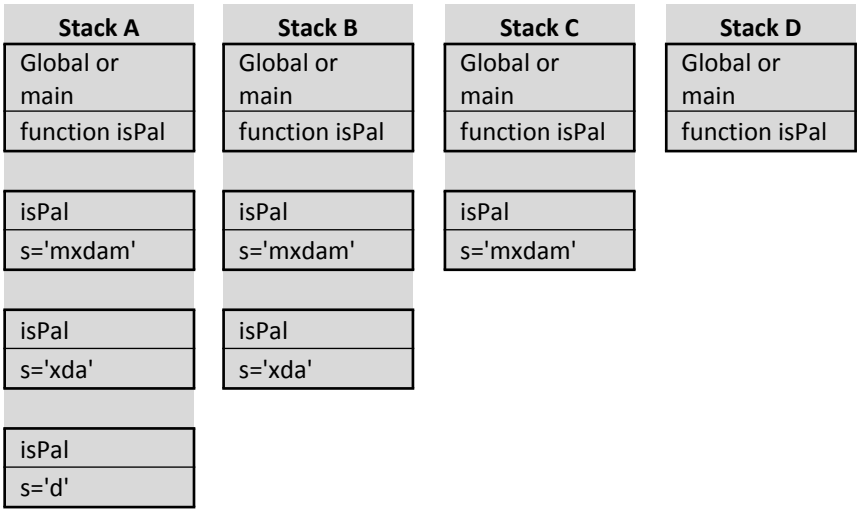
8. What will be the output of the following Python code?

```
x = 'abcd'
for i in range(x):
    print(i)
```

- a b c d
- 0 1 2 3
- … error …
- None of the above

9. Consider the following Python code to determine whether a given string is a palindrome. It uses recursion to do so. What is the **stack of frames <u>immediately after isPal is called for the LAST time, but before it returns</u>**. In other words, from those shown below, is it Stack A, Stack B, Stack C or Stack D. The format below is the same used by PythonTutor (note the stack grows downwards).

```
# Determine whether given string s is a palindrome
# Assumes s consists only of lower-case English letters
def isPal(s):
    if len(s) <= 1:
        return(True)
    return((s[0] == s[-1]) and isPal(s[1:-1]))
# test statement in main program
print(isPal("mxdam"))
```



- ○ Stack A
- ○ Stack B
- ○ Stack C
- ○ Stack D
- ○ None of the above

10. Consider the sample Python code given below. What are the variables that are local to **`func`**, and those that are global to **`func`**.

```
# Python code to understand which variables are accessible from within a function
def func(x, y):
    x = x + 1
    z = x + y
    return(z)

# test statements in main program
x = 4
y = 6
z = 12
print(x, y, z)
print(func(x, y))
```

- ○ Local to **`func`**: x, z; global to **`func`**: x, y, z
- ○ Local to **`func`**: x, y; global to **`func`**: z
- ○ Local to **`func`**: z; global to **`func`**: none
- ○ Local to **`func`**: z; global to **`func`**: x, y
- ○ None of the above

11. What will be the binary representation of 15.125?

- ○ 1.111001 * 2^-3 (base 2)
- ○ 1.111001 * 2^3 (base 2)
- ○ 1.111001 (base 2)
- ○ None of the above

12. What will be the value of **`count`** in the following code?

```
count = 0
for i in range(2,2,1):
    count = count+1
print(count)
```

- 0
- 1
- 2
- 3
- None of the above

13. I need to include a statement such as:
`k in [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23]:`
in my code. But this is too cumbersome, and difficult to change. I wish to instead use the "range" function to write an equivalent statement, such as:
`k in range(start, stop, step):`
What should that be?

- k in range(1, 23, 1)
- k in range(1, 23, 2)
- k in range(1, 24, 1)
- k in range(1, 24, 2)
- None of the above

14. A nested if-then-else statement produces an output that is dependent on the value of X and of Y. What is that if-then-else statement (written as pseudo-code) that produces an outcome given in Table below? **NOTE: a "-" in the table is to be interpreted as an empty string (or nothing).**

	Y < 0	Y >= 0
X < 0	yes	-
X >= 0	no	no

- If X < 0 and Y < 0, then print("yes")
  Else {if X >= 0 then print("no")
- If X < 0 then {if Y < 0 then print("yes")}
  Else print("no")
- If X >= 0 then print("no")
  Else {if X < 0 and Y < 0 then print("yes") }
- All of the above
- None of the above

15. What will be the output of the following Python code if `func3()` is called?

```
def func1(x):
    y = 5
    x = x + y
    return(x)
#
def func2(x):
```

```
    z = 6
    x = z - x
    return (x)
#
def func3():
    x = 9
    x = func1(x)
    func2(x)
    print(x)
#
func3()
```

- 14
- 9
- 8
- 3
- None of the above

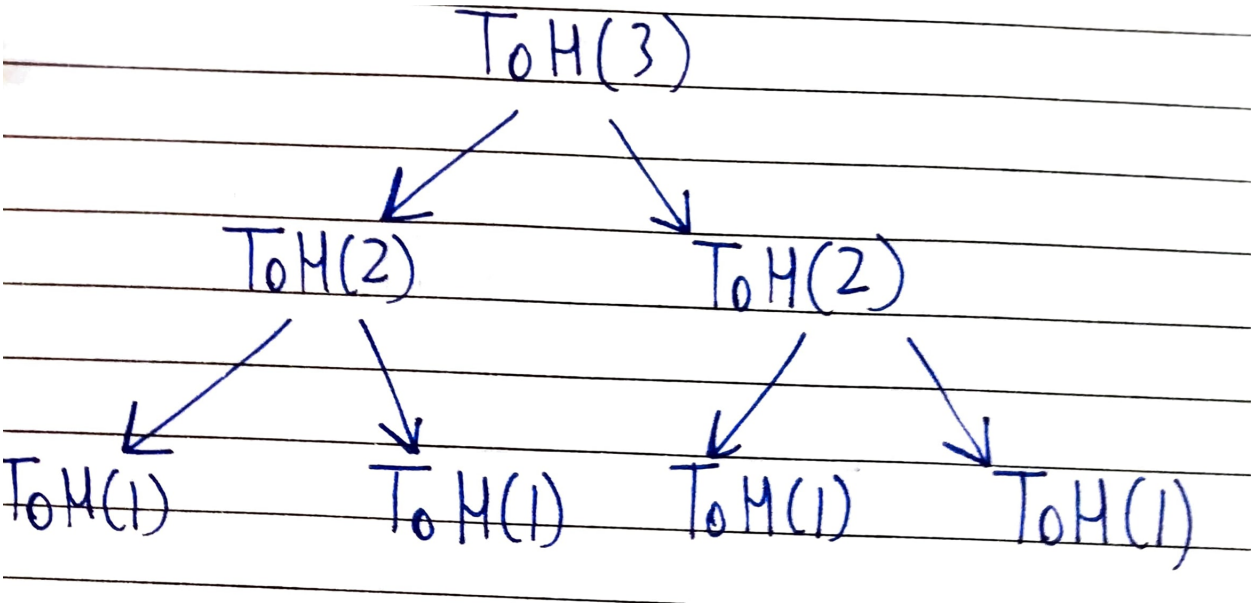
Mid-Sem MCQ Solutions

Q1.
Correct Answer: Line 4 and 5 only
Explanation: Strings are immutable, therefore we'll get an error if we try to modify a string.

Q2.
Correct Answer: print(L[1:8:2])
Explanation: This can be done using list slicing. In the desired output, the first element is 23 which is at index 1 in the original list and the last element is 75 which is the last element in the original list as well. So we'll slice from index 1 and will end at 8. It can also be observed that step value of the slicing is 2 because we are supposed to display alternate elements.

Q3.
Correct Answer: 5 6 7 8 9 10
Explanation: The loop will break when i%11=0. The current value of i is 5 and the nearest value for i when i%11=0 is i=11. So this code will display values of i from 5 to 10. When i becomes 11 the loop will break.

Q4.
Correct Answer: 7
Explanation: The recursive tree for the problem can be made as follows:



We can clearly see that 7 function calls are being made.

Q5.

Correct Answer: 5 16 g or Error

Explanation: The indentation was incorrect so the code will return an error. But ignoring indentation error, in the code before print(x) it is written global x. That means the global value of x will be used i.e 5. So it will display 5. Then f() is called, there we assigned x=6 (local variable this time). In the next line it is written global y, which means we are accessing the global variable y which holds value 10 currently. So print(x+y) will give 16. We then reassigned x as x="g" (this will modify the global x which initially was 5). print(x) will display 'g'. Therefore final output would be 5 16 g.

Q6.

Correct Answer: 11

Explanation: The initial value of i was 4. With every recursive call 'i' is getting decremented by 1 and j is getting incremented by 1. These recursive calls will be made until the base case is reached (i.e when i becomes 0). When i was 4, j was 7. When base case was hit, j will become 11 and this value of j will be returned by the function.

Q7.

Correct Ans: Once the code executes, resulting x will be 0, provided x > 2 initially

Explanation: In case x>2, the first if statement would be executed and it will double up the value of x making x>4. Since now x>4 the next if statement will get executed as well which will make the value of x as 0.

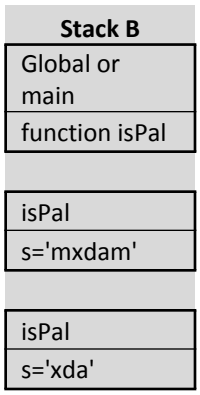
Q8.

Correct Ans: error

Explanation: Strings are not iterable hence it will throw an error.

Q9.

Correct Ans: Stack B



Explanation:

isPal() is a recursive function that initially takes the string madam, the 1st time the 1st and last letters are same('m') so the condition `(s[0] == s[-1])` in the return statement evaluates to true and the next recursive call is made starting from the 2nd index of "mxdam" till the end which is "xda". Now this time 'x' != 'a' , so the 1st condition in the return statement evaluates to False and no more recursive calls is made, since for AND if the 1st condition is False, that makes the entire result as False. The 2 stack frames are then popped out from the memory stack and returned to main

Q10.
Correct Ans: None of the above
Explanation:
x,y,z are defined in the functions as well , so they override the global scope of the operators x,y,z, if we wanted to use the globally defined variables x,y,z then we need to use the global keyword before the variable name in the function
E.g

```
def f():
    s = "Me too."
    print(s)

# Global scope
s = "I love programming"
f()
print(s)
```

---

o/p:

```
Me too.
I love programming.
```

```
def f():
    global s
    print(s)
    s = "Look for Python Section"
    print(s)

# Global Scope
s = "Python is great!"
f()
print(s)
```

---

o/p:

```
Python is great!
Look for Python Section.
Look for Python Section.
```

Q11.

Correct Ans: 1.111001 * 2^3 (base 2)

Explanation:

The binary of 15 is 1111 and that of 0.125 is 0.001 , so the result is 1111.001 , shifting to the left by 3 decimal places makes it 1.111001 *2^3

Q12.

Correct Ans: 0

Explanation:

range(2,2,1) indicates that the starting of the range is 2 and ends at (2-1=1) which is not possible since it is an increasing range, so the for loop never executes the line within it, so the count value is 0

Q13.

Correct Ans: k in range(1, 24, 2)

Explanation:

to print in steps of 2 as 1,3,...23, we provide start of range as 1 and end as 24 so it ends on 23 and the step value as 2

Q14.

Correct Ans: All of the above

Explanation: (the scope of 1 if block is mentioned by braces{} **)**

a if X < 0 and Y < 0,
then print(“yes”) #x<0 andy>=0 need not be checked explicitly since we dont need a result
Else if X >= 0
then print(“no”) # applicable for any value of y only if x>=0

Hence, Prints the yes,no in the table in the given format

b If X < 0 then
if Y < 0 then
print(“yes”)} #x<0 andy>=0 need not be checked explicitly since we dont need a result
Else print(“no”) # applicable for any value of y only if x>=0

Hence, Prints the yes,no in the table in the given format

c If X >= 0
then print(“no”) # applicable for any value of y only if x>=0
Else
if X < 0 and Y < 0 then

print(“yes”) #x<0 andy>=0 need not be checked explicitly since we dont need a result

Hence, Prints the yes,no in the table in the given format

Q15.
Correct Ans: 14
Explanation:
the code flow is as below:

main() calls func3()
In func3()
x=9
x=func1(9)

In func1(x):

y = 5
x = x + y
 =9+5=14
return(x) #returns 14

So now x in func3() is 14

Next func2(14) is called but the return value of func2() is not stored in x in func3(), so the value of x is func3() remains 14 , so when we print the value of x in func3() its 14

Therefore, ans :14