## COL1000 Mid-term Exam, 2025-26 Sem I (89 marks, 120 minutes)

**Kerberos ID** _ _ _ 25 _ _ _ _ **Name**

Re-enter the last digit of your Kerberos ID: 1 ➔ (This is your **LDIG**. You will need to use it in some questions below.)

**Read all instructions and questions thoroughly and carefully. No clarifications may be sought during the exam. Write your kerberos ID on every sheet.**

Please sign the pledge below: (Exam will not be graded without your signature)

**I will neither seek help with the answers nor provide help during this exam.**

_____

Note: There are 17 questions on 4 pages. [ ] lists marks available for each question.
Limit your answers to given spaces. Rough-sheet is included in the back. Do not detach it.
If a question seems ambiguous, write down your interpretations, and answer all variants.

1. [4] You have to visit a website via a browser, but it throws up the following error. Explain the error. Analyze the listed actions – what set of steps should you consider and under what condition?

> Someone could be trying to impersonate the site. You should not continue.
>
> Websites prove their identity via certificates. This browser does not trust mybankk.com because its certificate issuer is unknown.
>
> Error code: SEC_ERROR_UNKNOWN_ISSUER
>
> Go Back (Recommended) Accept the Risk and Continue

1. Meaning of the error code: Certificate presented by the website has been issued by a certificate authority which is not recognized by the browser.
2. Action: Go back and verify the website address. If incorrect, type the correct address.
3. Action: Continue and inspect certificate. Proceed if the certificate authority is verifiable independently of the browser. (e.g., using CA certificate)

2. [5] Given the following program, provide the requested values after the execution of each statement. Be sure to format lists and tuples appropriately.

| x = **LDIG** | | Value of: x | Value of: y |
|---|---|---|---|
| x += x | ➔ | 2 | |
| x, y = x+9/2, x+9//2 | ➔ | 6.5 | 6 |
| [lxy, txy] = list((x, y)), (x,y) | ➔ | lxy [6.5, 6] | txy (6.5, 6) |
| lxy2, txy2 = lxy, txy | ➔ | lxy2 [6.5, 6] | txy2 (6.5, 6) |
| lxy += lxy2 | ➔ | lxy [6.5, 6, 6.5, 6] | lxy2 [6.5, 6, 6.5, 6] |
| txy += txy2 | ➔ | txy (6.5, 6, 6.5, 6) | txy2 (6.5, 6) |

3. [4] What is the value of each of the following expressions? If the expression has an error, specify that error instead.

| | | |
|---|---|---|
| a) | True == False | ➔ False |
| b) | True == not False | ➔ SyntaxError (“True == not” is meaningless. Note that == precedes not.) |
| c) | not True == False | ➔ True |
| d) | 10 == 5+5 > False | ➔ True |

4. [5] Given the following program, provide the requested values after the execution of each statement. Do remember to format lists and tuples properly.

```
t1 = t2 = (0, 1, -LDIG)        # Inititalize t1 and t2
l1 = l2 =  [0, t1, LDIG]       # Initialize l1 and l2
t1 += t1[1:3]                  ➔  t1 (0, 1, -1, 1, -1)        t2 (0, 1, -1)
l1 = l1[1:3]                   ➔  l1 [(0, 1, -1), 1]          l2 [0, (0, 1, -1), 1]
l1.append(l2)                  ➔  l1 [(0, 1, -1), 1, [0, (0, 1, -1), 1]]   l2 [0, (0, 1, -1), 1]
l2 = [x*x for x in t2]         ➔  l2 [0,1,1]                  t2 (0, 1, -1)
l2 = [x*x for x in t2 if x !=0]  ➔  l2 [1,1]
l2 = [x*x if x >0 else -x*x for x in t2 if x  != 0]  ➔  l2 [1,-1]
```

5. [3] What is printed in each of the following programs? If there is an error, specify the error instead.

a)
```
l1, l1[0] = [(1,2,3), LDIG]
print(l1)
```
l1: TypeError. (Tuple cannot be mutated.)

b)
```
l1, l1[0] = ([1,2,3], LDIG)
print(l1)
```
l1: [1,2,3]

c)
```
name = 'Beta'
if name == 'alpha' or 'Alpha':
        name = 'ALPHA'
print(name)
```
name: ALPHA

6. [5] Consider each line of the following program in sequence. If there is an error on that line, given the earlier valid statements, specify the error. If there is no error due to that statement, write **None** instead.

| | | |
|---|---|---|
| value = [1] | ➔ | None |
| 2xvalue = value + [2] | ➔ | SyntaxError. (Name cannot begin with digit) |
| value = Value+[3] | ➔ | NameError. (Names are case-sensitive. Value is different from value.) |
| value[1] | ➔ | IndexError. (value is a list with one element, only index 0 is valid.) |
| value[0], value[0] = 3 | ➔ | TypeError. (Cannot unpack int on RHS to two items) |

7. [4] What is the result of executing each of the following programs? Specify the error instead, if there is one. Denote the space character by @ and end-of-line by $ in your output. Assume that the user input on prompt is **1,2** followed by End-of-line (i.e., *Enter*) each time.

a)
```
input1 = input('input1: ')
print('user' + input1)
```
➔ user1,2$

b)
```
input1 = input('input: ')
print('user', input1)
```
➔ user@1,2$

c)
```
input1 = input('input: ')
print('user', input1.split())
```
➔ User@['1,2']$

d)
```
input1 = input('input: ')
print('user' + input1.split())
```
➔ TypeError. (Cannot add string with list)

8. [6] Assume **days_attended** refers to the number of classes attended and **total_days** refers to the total number of classes held. **Mark** refers to the total marks obtained by some student. The Table below maps marks to grade, except if **days_attended** is less than or equal to 3/4th of **total_days**: the listed grade must be lowered by 1 (subject to 0 remaining the lowest possible grade). Write Python code that computes the grade **Grade** for some student, given **Mark**, **days_attended**, and **total_days.** (You may write the code in two columns, left half first.)

| | |
|---|---|
| marks ≥ 85 | grade = 10 |
| 85 > marks ≥ 70 | grade = 8 |
| 70 > marks ≥ 55 | grade = 6 |
| 55 > marks ≥ 40 | grade = 4 |
| 40 > marks | grade = 0 |

```
if marks >= 85: grade = 10
elif marks >= 70: grade = 8
elf marks >= 55: grade = 6
elif marks >= 40: grade = 4
else: grade = 0
grade -= 0 if grade == 0 or days_attended > 3*total_days/4+1E-9 else 1
---- Alternatively:
if days_attended < 3*total_days/4+1E-9 and grade > 0:
    grade -= 1
```

9. [5] What is printed by the following code? Explain.

```
ii = 5
while ii >= 1:
    jj = ii
    ii -= 1
print(jj)
for jj in range(jj, 5, 2):
    print(5+jj//2)
```

Output:

1
5
6

Explain:
The first loop continues until ii becomes <1, i.e., 0. jj refers to the value of ii before decrement, so it is 1. The second loop traverses range(1,5,2), which iterates to 1 and 3. The later modification of jj has no impact on the iteration count. Output are 5+1//2, and 5+3//2.

10. [6] Given int variables x and y, the following code categorizes cases. Does it cover all cases of possible values of x and y? If not, list the cases that are not covered. Does it have code that would never be executed (unreachable code) ? If so, provide the line number(s) and the specific part(s) of the line(s)?

```
L1: if(x < -1 or y < -2):
L2:     case = 0
L3: elif(x > 1 or y > 2):
L4:     if(x < y): case = 1
L5:     elif y < 1: case = 2
L6: else:
L7:     case = 3 if x+y > 3 else 4
```

Cases Not covered:
No case if (x > 1 or y > 2) and x >=y and y >= 1
y>2 is implied by y>=1. So: x>1 and y>=1 and x >= y suffices
Unreachable code:
case = 3 at L7
Because else reached if -1 <= x <= 1 and -2 <= y <= 2,
in this range x+y could be 3 but never >3.

11. [4]

```
a = LDIG
b = 5
while b > 1+b//2:
    a += LDIG
    b -= 2
```

Examine the program on the left.
➔ What values do a and b refer to at the end? a: 3 b: 1

12. [5]

```
nxt = n
while 2*nxt < 20*n:
    nxt += nxt
print(nxt)
```

Examine the program on the left, and provide the data requested below as an expression in n. Assume n is int.

| | |
|---|---|
| The number of times the loop iterates: | 4 if n > 0 else 0 |
| The value printed at the end: | 16n if n > 0 else n Alt: 16n if n>=0 else n |

13. [6] Given two lists of ints, namely list1 and list2, complete the following nested loop to compute a list of lists list3 such that list3[i] is the list of unique elements of list1, which are divisible by list2[i].

Example:
list1: [10,20,20,35] list2: [2,5]
→ list3: [[10, 20], [10, 20, 35]]

```
list1.sort() # sort list1 in increasing order
List3 = []Alt: list3 = [[] for e in list2]
for i in range(len(list2)):
    list3.append([])    Alt: pass

    Prev = None
    for e in list1:                          # For elements of list1
        if e != Prev and not e%list2[i]:     # Condition that e may be added to list3

            list3[i].append(e)               # Add to appropriate list in list3 (Indent properly)

            Prev = e
```

14. [8] Examine the given program. What are the possible output, considering all input cases, including any corner cases? List the different possible output and all input which result in that output.

```
R = int( input('R '))
Done = None
for r in range(R, 2*R, R//2):
    if r >= 2*R-1:
        Done = 1
if(Done): print(Done)
```

(In case there is no output, list **None** in the output box. If there is an error instead, specify that error. In the Case box, list the corresponding input.)

| | | | |
|---|---|---|---|
| Output | ValueError | Case I | Input is not an integer |
| Output | ValueError | Case II | Input is 0 or 1, range does not exist |
| Output | 1 | Case III | Input is odd (but not 1) or -ve or 2 |
| Output | None | Case IV | Input is positive even > 2 |

15. [5] Point out the logical error(s) in the following program, which seeks to compute the factorial of an integer without using the * operator. Also fix those error(s). You may refer to the line numbers shown.

```
L1: n = int(input('Input an integer: ')) # Assume user enters an integer
L2: if n < 0: print('Factorial of negative integers are not defined')
L3: factorial = 0
L4: for i in range(n):
L5:     Sum = 0
L6:     for j in range(i):
L7:         Sum += factorial
L8:     factorial = Sum
L9: print(f'Factorial of {n} is {factorial}')
```

Errors:
0! must be 1. Multiplication with initial 0 would yield 0.
Outer loop computes i! from (i-1)!: It must go from 1 to n.

Fix:
L3: factorial = 1
L4: for i in range(1, n+1)

16. [7] Complete the following program to insert element e at index i of list L. Before insertion, each element at position i and greater must be shifted one position higher (i.e., L[j] is moved to L[j+1]) to make space for e. Make sure that no elements of L are lost in the shifting process.

```
L.append(None)  Alt: L.append(L[-1])          # Any initialization
for j in range(len(L)-1,i+1,-1):   Alt: Start len(L)-2.# Iterate over positions of the list
    L[j] = L[j-1]                             # Move element of L to make space
L[i] = e
```

17. [7] Provide the output of the following program.

```
list1 = [[i*i for i in range(j)] for j in range(5)]
print(list1)
sum = 0
for le in list1:
    for i in range(1, len(le)):
        if le[i]-le[i-1] > 1:
            sum += le[i]
            break
        print(sum, end='#')
    print(sum, end=',')
    if sum > 1:
        break
print(f'\nSum is {sum}')
```

```
[[], [0], [0, 1], [0, 1, 4], [0, 1, 4, 9]]
(jth list has square of elements in range(j)
0,0,0#0,0#4,
(inner break avoids #, ends only inner loop)
Sum is 4
```