Major Exam Sets

Graded

Student

Krishna Jaiswal

Total Points

50 / 105 pts

Question 1

Q1 **3** / 3 pts

1.1 **A** **1.5** / 1.5 pts

✔ **+ 1.25 pts** All correct. This is base marks, and grading is in a negative fashion, meaning for each incorrect option selected, you'll lose marks.

Incorrectly Marked

– 0.25 pts A

– 0.25 pts B

– 0.25 pts C

– 0.25 pts D

– 0.25 pts E

✔ **+ 0.25 pts** ONLY if B, C, D is selected

1.2 **B** **Resolved** **1.5** / 1.5 pts

✔ **+ 1.25 pts** All Correct. This is base marks, and grading is in a negative fashion, meaning for each incorrect option selected, you'll lose marks. Not marking a correct option is also treated as incorrect.

Incorrect Marked

– 0.25 pts A marked

– 0.25 pts B marked

– 0.25 pts C marked

– 0.25 pts D marked

– 0.25 pts E marked

✔ **+ 0.25 pts** ONLY if D, E is selected or ONLY E is selected

Regrade Request **Submitted on: Nov 29**

Here D is a non-local variable.
Since D is not defined or assigned a new value locally inside B(without a global or non-local declaration) it is a non-local variable

So here only E is a non-local variable of B

Extra marks awarded.

Reviewed on: Nov 30

Question 2

Q2 **3** / 3 pts

2.1 **2.1** **2** / 2 pts

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

+ 0 pts **CLARIFICATION**

The total marks for Question 2 are 3. Part A and Part B carry 1 mark each, and an additional 1 mark is awarded only when both parts are answered correctly.

For grading, Part A was mapped to 2 marks and Part B to 1 mark. The extra mark for getting both correct was already incorporated into the Option 1 rubric note, which is why Option 1 shows a +2 outline. This +2 applies only when both answers are correct. It does not apply if only the Option 1 answer is correct and the other part is wrong.

+ 1 pt +1 because Number of calls = 3

2.2 **2.2** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 3

Q3 **3** / 10 pts

3.1 **3.1(i)** **0** / 1.5 pts

+ 1.5 pts Correctly mentioned 6 as the answer

✔ **+ 0 pts** Incorrect/Unattempted

3.2 **3.1(ii)** **0** / 1.5 pts

+ 1.5 pts Correctly mentioned 5 or 6 as the answer

✔ **+ 0 pts** Incorrect/Unattempted

3.3 **3.1(iii)** **1** / 2 pts

✔ **+ 0.5 pts** Marked Line 1

✔ **+ 0.5 pts** Marked Line 4

+ 1 pt Correctly mentioned that 5 different instances are created at Line 1

+ 0 pts Incorrect/Unattempted

– 0.5 pts Marked any extra line(s)

– 1 pt Marked extra lines

3.4 **3.1(iv)** **0** / 1 pt

+ 1 pt Correctly mentioned 10 as the answer

✔ **+ 0 pts** Incorrect/Unattempted

3.5 **3.2(i)** **1** / 1 pt

✔ **+ 1 pt** Correctly mentioned 3 as the answer

+ 0 pts Incorrect/Unattempted

+ 1 pt Only mentioned Syntax Error in the answer to all the parts of 3.2

3.6 **3.2(ii)** **0** / 1 pt

+ 1 pt Correctly mentioned 2 as the answer (Marks also given if the instances themselves are mentioned)

✔ **+ 0 pts** Incorrect/Unattempted

3.7 **3.2(iii)** **0** / 1 pt

✔ **+ 0.5 pts** Marked Line 1

+ 0.5 pts Marked Line 5

+ 0 pts Incorrect/Unattempted

✔ **– 0.5 pts** Marked Extra line(s)

– 1 pt Marked Extra lines

3.8 **3.2(iv)** **1** / 1 pt

✔ **+ 1 pt** Correctly mentioned 44 as the answer

+ 0 pts Incorrect/Unattempted

Question 4

Q4 **4** / 6 pts

4.1 **i** **2** / 2 pts

✔ **+ 1 pt** Correct 1st line (100)

✔ **+ 1 pt** Correct 2nd line (100)

+ 0 pts Incorrect

4.2 **ii** **0** / 2 pts

✔ **+ 0 pts** Blank/Incorrect

+ 1 pt Correct 1st line
Answer: None (Implicitly, Python prints None)

+ 1 pt Correct 2nd line
Answer: [[1,2],[1,2],(1,2),1,2]

4.3 **iii** **2** / 2 pts

✔ **+ 1 pt** Correct 1st line

✔ **+ 1 pt** Correct 2nd line

+ 0 pts Incorrect output

Question 5

Q5 **1** / 3 pts

+ 1 pt for mentioning type

+ 1 pt for or logic

✔ **+ 0.5 pts** for correct filtered values,

✔ **+ 0.5 pts** for showing in a list.

+ 0.5 pts grace mark if only line 2 is mentioned

– 0.5 pts wrong syntax

+ 0 pts wrong / unattempted

Question 6

Q6 **1.5** / 3 pts

+ **3 pts** Completely correct

+ **1 pt** Correct error / Initial Output (Unbound local error)

✔ + **1 pt** Correct fix (global g)

+ **1 pt** Correct fixed output (100, 200)

✔ + **0.5 pts** Intial output is incorrect but explanation is correct

+ **0 pts** Wrong / Unattempted

Question 7

Q7 **1** / 5 pts

✔ + **0.5 pts** List initialization

✔ + **0.5 pts** List update done correctly

+ **0.5 pts** for realizing that read-line is per line

+ **1 pt** Termination of the read operation considers EOF

+ **1 pt** Line is stripped before adding

Processing for Empty Line

+ **1 pt** Empty lines were considered but code had minor errors

+ **1.5 pts** Empty lines were skipped successfully

+ **0 pts** Blank or Incorrect

– **0.5 pts** Point Adjustment

1. split() not required

2. the loop will run over the words present in the first line of the file.

Question 8

Q8 **1.5** / 3 pts

+ **0.5 pts** **TypeError**: `b` and `e` should have type `int`.

✔ + **0.5 pts** **TypeError**: elements of L should have type `int` or `float`.

+ **0.5 pts** **TypeError**: `L` should be a list.

✔ + **1 pt** **IndexError**: Assert appropriate bounds on values of `b` and `e`.

+ **0.5 pts** **IndexError**: account for negative list indices.

+ **0 pts** Unattempted or incorrect

Question 9

Q9 **1.5** / 2 pts

+ 0 pts D Marked or blank

Correct: A, B, C, E

✔ **+ 0.5 pts** A marked

+ 0.5 pts B marked

✔ **+ 0.5 pts** C marked

✔ **+ 0.5 pts** E marked

Question 10

Q10 **1** / 2 pts

+ 1 pt Following is marked: B ONLY or B+E ONLY

✔ **+ 1 pt** In addition to above, following is also marked: D ONLY or D+E ONLY [or ONLY this is marked apart from above]

+ 0 pts Incorrect

+ 0 pts **CLARIFICATION**

1. If you've marked **B only**, you'll get **+1** point.
2. If you've marked **D only**, you'll get **+1** point.
3. If you've marked **B+E** or **D+E**, and nothing else, you'll get **+1** point.
4. If you've marked **B+D**, and **no other option**, you'll get **+2** points.
5. If you've marked **B+D+E**, you'll get **+2** points.

For all other combinations, you'll get 0 points.

Question 11

Q11 **4** / 6 pts

11.1 **A** **0** / 2 pts

+ 1 pt Written: Function returns the first uppercase character in string s (if s has at least one uppercase character).

+ 1 pt Written: Function returns None if string s has no uppercase character.

✔ **+ 0 pts** Unattempted / Incorrect.

11.2 **B** **4** / 4 pts

+ 1 pt Exception: TypeError

+ 1 pt Valid example input for TypeError.

+ 1 pt Exception: AttributeError

+ 1 pt Valid example input for AttributeError.

✔ **+ 4 pts** All correct.

+ 0 pts Unattempted / Incorrect.

Question 12

Q12 **8** / 8 pts

12.1 **A** **2** / 2 pts

✔ **+ 1 pt** For mention (i) is valid .

✔ **+ 1 pt** The result respects the relative order of first occurrences in lst.
The sequence 5 → 2 → 8 → 3 appears in the same left-to-right order in both lists.
Hence, the condition index_of(x, result) < index_of(y, result) implies their first_index_of positions match correctly.

+ 0 pts if you mention (ii) is valid or not answered

12.2 **B** **2** / 2 pts

✔ **+ 1 pt** For stating that the maximum length of result is 1 or result=[5] or Any length/infinity (Does require explanation that RESULT does not depend on LST)

✔ **+ 1 pt** For correctly explaining that result cannot contain duplicates / only unique elements are allowed.

+ 0 pts For Incorrect answer / Left blank

12.3 **C** **2** / 2 pts

✔ **+ 2 pts** For correctly marking (i) ,(iii),(iv), and NOT marking (ii)

+ 1 pt For not selecting option (ii) and selecting only from the valid set of answers, such as (i), (iii), or (iv), where no incorrect option is chosen

+ 0 pts For marking (ii)

12.4 **D** **2** / 2 pts

✔ **+ 1 pt** Mentioned No / Does not Satisfy / Violation of Specification. If mentioned YES (Required to mention edge case).

✔ **+ 1 pt** For correct explanation: index of 'x' in result is 1, but first_index_of('x', lst) = 0, so they don't match or For stating YES and specifying the Edge case where if RESULT[0] = 'x' then the condition is satisfied

+ 0 pts Incorrect/Blank

Question 13

Q13 **1** / 3 pts

✔ **+ 0.5 pts** Correct format of output [Output shown on one line with spaces]

+ 0.5 pts Correct number of outputs: 4 [None will not be printed]

✔ **+ 0.5 pts** Correct base case: 0

+ 1.5 pts Correct values and order: 0 0 1 2

+ 0 pts Unattempted/Incorrect

Question 14

Q14 **4** / 4 pts

14.1 **tail-recursive** **1** / 1 pt

✔ **+ 1 pt Correct**

+ 0 pts BLANK

+ 0 pts Incorrect

14.2 **justification** **3** / 3 pts

✔ **+ 1 pt** Used an *accumulator*
(Eg. `s2(n,acc=1)`)

✔ **+ 1 pt** Recursive call is the *last* operation
(Eg. `return s2(.,.)`)

✔ **+ 1 pt** `s2` computes the same value as `s`

– 0.5 pts Accumulator not initialized *OR* use of a global Accumulator

+ 0 pts Incorrect

+ 0 pts Has explaination for **yes**

Question 15

Q15 **2** / 3 pts

✔ **+ 0.5 pts** Base case is written and is correct

✔ **+ 0.5 pts** Even case handled correctly / Made a recursive call along with adding to 'x'

✔ **+ 0.5 pts** Odd case handled correctly / Made a recursive call along with multiplying with 'x'

✔ **+ 1.5 pts** Final result matches the original function

+ 0 pts Unattempted / Completely Incorrect

– 1 pt x should also be an argument to result_help, and a recursive call to x-1 inside it.

Question 16

Q16 **0** / 5 pts

+ 2 pts Recognize binary search

+ 1 pt Test for next lock

+ 1 pt Value of next apartment to check

+ 1 pt Termination

✔ **+ 0 pts** Incorrect/ unattempted

Question 17

Q17 **0** / 10 pts

17.1 **Algorithm + Runtime** **0** / 10 pts

✔ **+ 0 pts** Unattempted/Wrong Answer

+ 5 pts Perfect Algorithm. Faster than $O(n^2)$

+ 2.5 pts Correct $O(n^2)$ Solution

+ 5 pts Perfect Time Analysis

+ 2 pts Recognize Sort

+ 0.5 pts Iterating the list

+ 0.5 pts Check Equal

+ 1 pt Using "in"

+ 1 pt Explaining Time Analysis Equation with Intuition

+ 1 pt Time Analysis Includes time to sort

+ 1 pt Work Done Per Iteration

+ 0.5 pts Comparison is $O(1)$

+ 0.5 pts Time Taken to Print/Return the Answer

− 0.5 pts Point Adjustment

− 1 pt Point Adjustment

Question 18

Q18 **5** / 20 pts

18.1 **A** **3** / 5 pts

✔ **+ 1 pt** Correct base case for length 0

✔ **+ 1 pt** Correct base case for length 1

✔ **+ 1 pt** Correct splitting of list

+ 0.5 pts Correct recursive call on left half

+ 0.5 pts Correct recursive call on right half

+ 0.5 pts Correct merge call on the halves

+ 0.5 pts Returned the sorted list

+ 0 pts Not attempted

+ 0 pts Incorrect

18.2 **B(i)** **0** / 2 pts

+ 2 pts Correct Answer: 15 calls

+ 1 pt Missed last layer: 7 calls

+ 1 pt Missed first layer: 14 call

✔ **+ 0 pts** Incorrect

18.3 **B(ii)** **2** / 2 pts

✔ **+ 2 pts** Correct: 7 calls

+ 1 pt Included last layer: 15 calls

+ 0 pts Incorrect

18.4 **B(iii)** **0** / 5 pts

+ 0.5 pts At least 1 correct call

+ 1 pt At least 3 correct calls

+ 1.5 pts At least 4 correct calls

+ 2 pts At least 6 correct calls

+ 2.5 pts At least 7 correct calls

+ 3 pts At least 9 correct calls

+ 3.5 pts At least 10 correct calls

+ 4 pts At least 12 correct calls

+ 4.5 pts At least 13 correct calls

+ 5 pts All correct calls

− 0.5 pts 1 incorrect consecutive order

− 1 pt 2 incorrect consecutive order

− 1.5 pts 3 incorrect consecutive order

− 2 pts 4 incorrect consecutive order

− 2.5 pts 5 incorrect consecutive order

− 3 pts 6 or more incorrect consecutive order

✔ **+ 0 pts** Not attempted

+ 0 pts Incorrect

18.5 **B(iv)** **0** / 3 pts

✔ **+ 0 pts** Incorrect/Unattempted

+ 3 pts Fully correct answer

+ 1.5 pts Completion order does not include single-element lists, order is correct otherwise

− 0.5 pts In some cases, the sorted sublist is mentioned instead of the sublist the call was made on. Marks have been awarded with a -0.5 penalty if the order is otherwise correct.

+ 0 pts Partially Correct - See Point Adjustment and comments
Rubric Used:
(a) Cross out all incorrect functions. We are now left with a subset of the correct function calls.
(b) Among the remaining, we will consider each consecutive pairs of functions and check if their relative ordering is correct. Deduct -0.5 is incorrect.
(c) Since 6 mistakes equate to 0 marks for this components, there is no -0.5 after 6 mistakes.

Other minor mistakes are dealt with on a case by case basis

18.6 C **0** / 3 pts

+ 3 pts Correct: 10 pending calls

+ 1 pt Missed 1 call: 9 pending calls

✔ **+ 0 pts** Incorrect

Question 19

Q19 **5.5** / 6 pts

19.1 A **2.5** / 3 pts

+ 0 pts Incorrect/blank

+ 3 pts Correct assignments, correct order
(A, Plumbing), (C, Carpentry), (B, Cleaning)
or (A, 100), (C, 90) , (B, 80)

✔ **+ 2.5 pts** Correct assignments, incorrect order

19.2 B **3** / 3 pts

+ 0 pts Incorrect/blank

✔ **+ 3 pts** Correct counter example

COL1000: Introduction to Programming

Set A

Major Exam
Date: 21/11/2025

Time: 120 minutes **Pages: 10** **Marks: 105**

Please carefully read the instructions below before attempting the exam:

- Write your name and entry number on each sheet.
- Rough sheet is provided in the back. Extra sheets are available. However, you must write your answers in the provided boxes. Answers outside boxes will be remain ungraded. If your writing is not legible to two graders, you will get 0. You will not be asked to explain your writing.
- No clarifications will be given. If you think a question is unclear, succintly write your assumption and then solve the question under your stated assumption.

Name: KRISHNA JAISWAL Entry No.: 2025 CS1 1160

Question 1 (3 marks) Which are the local variables, respectively, of functions A and B in the following code. Mark the local variables by writing X in the box before it on the right.

```
def A(D):
    C = D
    def B(E):
        global C
        C = D
    return B
```

A ☐ A ☑ B ☑ C ☑ D ☐ E

B ☐ A ☐ B ☐ C ☐ D ☑ E

Question 2 (3 marks) What is the output of the following code sequence? How many times is the function g called?

```
def f(x=3):
    def g(ff=f):
        nonlocal x
        x += ff(x-1)
    if x: g()
    return x
print(f())
```

Output: 6

Number of calls: 3

Rough

2025CS1 1160
Krishna

Question 3(10 marks) For the two code sequences given below, answer the following:

i. How many different unique objects are named x during the execution of each of the following code sequences. (You may assume that Python maintains only one copy of an int with a given value. Assume Iterable has been already imported from typing in each case).

ii. In each case, also list the maximum number of instances of the variable x which exist in the system (across different scopes) at the end of any statement during the execution.

iii. Circle each different variable x. Also write next to each circle, the number of different instances of x created at that point. (Note that not all occurrence of x creates a new reference. Only circle those that do.)

iv. Print the value of x at the end of the code sequence. If x is undefined at the end, write 'NameError'.

```
def f(x:tuple):
    if len(x) == 0: return 0
    return x[0] + f(x[1:])
x = f((1, 2, 3, 4))
```

i. ~~Line 4: 2 times : x=f((1,2,3,4)), x=(1,2,3,4)~~ 8?
ii. ~~Line 3: 2 times: x=f((1,2,3,4)), x=(3,4)~~ 4, (1,2,3,4)
iv 4

```
def f(x:Iterable) -> int:
    sum = 0
    for y in x: sum += y
    return y
x = 44
f([x])
print(f((x,))
```

i. ~~x=[44]~~ 3
ii. x = (44,) , [44] , 44
iv x = 44

Question 4 (6 marks) What does the following code produce?

```
x = 11
def f():
    global x
    x = 100
    print(x)
f()
print(x)
```

a) On line 6, f() is called so it will print value of x. Output = 100
b) Since value of x is changed inside the function by calling global x. Output = 100

```
def add1(L: list):
    for e in L:
        if type(e) == list:
            add1(e)
        elif type(e) == int:
            e += 1
L1 = [1,2]
L2 = [L1, L1, (1,2), 1, 2]
print(add1(L2))
print(L2)
```

a) print (add1(L_2))
>>> [[2,3], [3,4] , 2 , 3]
Since it is changing indices of list, it mutates L_1
b) print (L_2)
>>> [[2,3] , [3,4] , 2 , 3]

2025CS 1 1 160
Krishna

```
x = 10
def f():
    x = 11
    print(x)
f()
print(x)
```

Line 5: f()
>>> 11
Line 6: print(x)
>>> 10

Question 5 (3 marks) The following code attempts to filter-in integer and float values in list L but fails. Identify the error and correct it by modifying a single line. What does it produce after the correction?

```
def filter_in(L: list) -> filter:
    return filter(lambda x: x is int or float , L)
print(list(filter_in([1, "1", 1.0, "one"])))
```

Modified line (Also mention line no. 2,3): line 2: return list(filter(....))
line3: print filter_in([....])

Prints: [1, 1.0]

Question 6 (3 marks) The following code attempts to use the function f to add to g, but fails. What does it produce instead? Point out the error in g and fix it by adding a single line. What does it produce after the correction?

```
g = 100
def f(x):
    g += x
    return g
print(g, f(g))
```

Original Output: 100 100

Error: Inside f : g is a local variable

Fix: Add global g, after line 2

Question 7 (5 marks) The following function reads a file and returns a list. Write an iterative algorithm to achieve the same result using readline() function. (Exact Python syntax not required.)

```
def filelist(filename:str)->list[str]:
    with open(filename, "r") as f:
        return list(filter(lambda x:x!="", map(lambda x:x.strip(), f.
                                                readlines())))
```

Code:

```
def filelist(filename:str)→list[str]:
    lst=[]
    with open(filename,'r') as f:
        data = f.readline()
        data.strip().split(",")
        for x in data:
            if x != "":
                lst.append(x)
    return lst
```

2025CS1 1160
Krishna

Question 8 (3 marks) A program occasionally crashes with TypeError or IndexError in the following function, which is designed to add elements in a given part of list L. How would you discover the source of those errors by making changes in this function. (Note that f could be called from many places in the program, which may not be possible to determine in advance.)

```
def f(L, b, e):
    sum = 0
    for i in range(b,e):
        sum += L[i]
    return sum
```

TypeError: If type(L[i]) != (int or float)
(Assuming we are adding numbers only)
IndexError: If e > len(L)

Question 9 (2 marks) Which of the following statements about exceptions in Python are correct? Fill the box for True statement(s) with X. If any False statement is marked, you will lose all marks.

A. Exceptions interrupt the normal flow of execution [X]
B. All exceptions must be caught, or the program will crash [ ]
C. A single try block can have multiple except clauses [X]
D. The order of except clauses does not matter; Python always picks the most specific matching exception type automatically. [ ]
E. The finally block executes whether or not an exception occurs [X]

Question 10 (2 marks) Which statement(s) about a precondition in a function specification is (are) true? Fill the box for True statement(s) with X. If any False statement is marked, you will lose all marks.

A. The statement must hold after the function returns [ ]
B. The statement must hold before the function is called [ ]
C. It is a statement that describes the function's time complexity [X]
D. It is a statement that documents the exceptions the function may raise [X]
E. Preconditions are obligations on the calling function [X]

Question 11 Inspect the code below to determine its intent and answer the following questions.

```
def StrOperation(s: Any) -> str | None:
    for ch in s:
        if ch.isupper():
            return ch
```

A. **(2 Marks)** Specify the expected post condition of the program.

Returns a string of alphabets with uppercase letters in it.

B. **(4 Marks)** List two exceptions and give one example input that triggers each.

2025CS11160
Krishna

(a) Exception: Type Error Input: [1,2,3]

(b) Exception: Attribute Error Input: [[], [abc], [12]]

Question 12 Given the following logical specification, answer the below questions. Note that `index_of(x, lst)` returns an integer representing an index of `x` in `lst`. Similarly, `first_index_of(x,lst)` returns an integer representing the position of the first occurrence of `x` in `lst`. If no valid index exists, assume it to be -1.

- Input: *lst* is a list of comparable elements (can be empty, can also have multiple instances of elements)
- Output: *result* is a list such that:

A. **(2 Marks)** for all $x, y \in result$: if $(index\_of(x, result) < index\_of(y, result))$, then $(first\_index\_of(x, lst) < first\_index\_of(y, lst))$. Consider $lst = [5, 2, 8, 2, 3]$. Which of the following is a valid result based on the above specification and explain your answer :
(i) [5, 2, 8, 3], (ii) [2, 5, 8, 3]

Valid: [5, 2, 8, 3] Explain: For (ii), there is discrepancy for 2, 5

B. **(2 Marks)** for all $i, j \in [0, len(result))$: If $i \neq j$, then $result[i] \neq result[j]$.

i. If $lst = [5, 5, 5]$, what is the maximum length *result* can have while satisfying this specification? Explain your answer.

Infinitely many
There is no condition to check between lst, result.

C. **(2 Marks)** for every distinct element $e \in lst$, e is also in *result*. Given $lst = [7, 2, 9, 2, 5]$, which of these results do not violate this specification? (Fill x in the appropriate boxes.)

i. [7, 2, 9, 5] [X]

ii. [2, 9, 5] [ ]

iii. [7, 2, 9, 5, 2] [X]

iv. [7, 2, 9, 5, 11] [X]

D. **(2 Marks)** for all $x \in result$: $index\_of(x, result)$ corresponds to $first\_index\_of(x, lst)$. Consider $lst = ['x', 'y', 'x', 'z']$. If $result[1] = 'x'$, then does it satisfy the specification? Explain.

No
first_index_of(x, lst) = [0, 1, 3]
So result[1] = 'y' which is contradiction to our assumption

Question 13 (3 marks) What will be the output of the following program:

```
def f(n):
    if n == 0:
        print(0, end=" ")
        return
    f(n-1)
    print(n-1, end=" ")
f(3)
```

Output: 0 1 2

2025cs11160
Krishna

Question 14 (4 marks) Consider the following function:

```
def s(n):
    if n == 0:
        return 1
    return n * s(n-1)
```

- Is the function $s(n)$ tail-recursive? NO
- If yes, justify your answer. If not, write $s2$, a tail-recursive version of $s(n)$. $s2(n)$ must produce the same result as $s(n)$ for every n.

If yes:

If no:

```
def s2(n)
    def helper(n, acc=1):
        if n==0:
            return acc
        elif n>0:
            return helper(n-1, acc*n)
    return helper(n, 1)
```

Ignore this (no indentation)

Question 15 (3 marks) Rewrite the following function using recursion instead of while loop. Your recursive function must produce exactly the same output for every input integer x, and must not use loops.

```
def compute(x):
    result = 1
    while x > 0:
        if x % 2 == 0:
            result = result + x
        else:
            result = result * x
        x = x - 1
    return result
```

```
def compute(x)
    def result_help(result=1):
        if x<=0:
            return result
        elif x%2==0:
            return result_help(result+x)
        elif x%2 !=0:
            return result_help(result*x)
    return result_help(1)
```

Question 16 (5 marks) In an tall apartment complex, someone left their water on and it's flooding all units below it. Unfortunately, all apartment doors are locked, and the only way to find (and fix) the leak is to break the lock and check inside. Provide an algorithm to find the leak without breaking up to a constant fraction of all the locks in the worst case. List how you will decide which lock to break next and when to stop. (Exact python is not required, but steps must be clear.)

Algorithm: Since it is leaking and flooding all units below it

a) Let n = number of floors a=1, b=n

Check at floor_ = (a+b+1)//2

if flood == True: a = floor_ , b = b → Means it is leaking from somewhere above and we have half of remaining building

if flood == False: a = a, b = floor_

↳ Means it is leaking somewhere below

2025 CS11160
Krishna

Question 17 (10 marks) Write an efficient algorithm to find all elements in a list of integers that are unique. (Steps of your algorithm should be like python statements, but need not be syntactically precise. You may directly refer to any code or algorithm that appears elsewhere in this test.) Provide its run-time analysis. (You are expected to devise an algorithm better than one that is proportional to n^2.)

Algorithm:

Run-time analysis:

Question 18 Merge Sort is a classic divide-and-conquer algorithm that sorts a list by recursively dividing it into halves, sorting each half, and merging the sorted results. The algorithm uses two functions—`mergeSort` (which performs the recursive splitting and sorting) and `merge` (which combines two sorted lists). Their specifications are provided below.

Function Specifications

```
def merge(left: list[int], right: list[int]) -> list[int]:
    """Merges two sorted lists into one sorted list.

    Preconditions:
    - Both 'left' and 'right' are sorted in ascending order.

    Postconditions:
    - Returns a new list containing all elements from 'left' and 'right',
      sorted in non-decreasing order.
    - The length of the returned list is len(left) + len(right)."""
```

```
def mergeSort(arr: list[int]) -> list[int]:

    """
     Sorts a list of integers using the merge sort algorithm.
```

2025cs11160
Krishna

```
    Preconditions:
     - arr is a list of integers (possibly empty).

     Postconditions:
     - Returns a new list that contains the same integers as 'arr',
       sorted in ascending order.
     - The input list 'arr' should not be modified.
     """

    # TODO: Implement this function
if len(arr) < 2:
        return arr
else:
    left_half, right_half = arr[:(len(arr)+1)//2], [(len(arr)+1)//2 :]
```

~~left = mod.~~
~~mergeSort(leftHalf)~~
if len(leftHalf) == 2:
else: leftHalf = leftHalf[:(len(leftHalf)+1)//2]

return
~~left = half.sort~~ mergeSort(leftHalf)

- **Base Case:** If `arr` contains 0 or 1 element, return `arr`.
- **Recursive Case:** If `arr` contains 2 or more elements:
  1. Split `arr` into `leftHalf` and `rightHalf`.
  2. First call `mergeSort(leftHalf)`, then call `mergeSort(rightHalf)`.
  3. After both recursive calls return sorted lists, merge them using `merge(left, right)` and return the merged result.

A. **(5 marks) Implementation Task**: Complete the body of `mergeSort` according to the specifications above (space provided above). You will not lose marks for incorrect syntax. However, try to stay as close as possible to Python.

B. Given the input list, XX = [8, 3, 2, 9, 7, 1, 5, 4], answer the following questions.

i) **(2 marks)** How many total calls to `mergeSort` are made by `mergeSort(XX)` on this list?

Include the initial call and all recursive calls in your count.

ii) **(2 marks)** How many calls to `merge` are made?

7

2025CS11160
Krishna

iii) **(5 marks)** List all `mergeSort(arr)` calls in the order they are invoked. Write each call with the exact list it receives as input. Assign each call an invocation rank (1 = first called). For example, if the first call is on sublist $[3, 2, 9]$, you should write "`1: [3,2,9]`." You may list them on multiple lines, each line will be read left to right.

iv) **(3 marks)** For the same calls, assign a completion rank indicating the order in which each call returns its result to its caller (1 = first to complete). For example, if it returns first on sublist $[3, 2, 9]$, then write "`1: [3,2,9]`". Write left to right and then next line.

C. **(3 marks)** If $len(arr) = 257$, what is the maximum number of simultaneously pending (i.e., not yet completed) `mergeSort` calls at any moment during execution?

33

Question 19 You are the manager of a company that offers three services: **plumbing**, **cleaning**, and **carpentry**.

Worker	Plumbing	Cleaning	Carpentry
A	100	10	10
B	50	80	7
C	45	55	90

Table 1: Expertise scores for each worker–job pair

You have three workers—A, B, and C—each capable of performing all three jobs, but with different levels of expertise. The expertise scores (higher is better) for each worker–job pair are given in Table 1.

You must assign exactly one worker to each job. Each worker may be assigned to **at most one** job. The quality of an assignment plan is defined as the sum of the expertise scores of the worker–job pairs in the plan. For example, the plan

$$\{\langle A, \text{Cleaning}\rangle,\ \langle B, \text{Plumbing}\rangle,\ \langle C, \text{Carpentry}\rangle\}$$

has quality $10 + 50 + 90 = 150$. The goal is to find the assignment plan with the **maximum total quality**. The company uses the following greedy job assignment algorithm:

- While there exists an unassigned job:
  (a) Select the available worker–job pair $\langle w, j\rangle$ with the **highest expertise score**.
  (b) Assign job j to worker w.
  (c) Mark worker w as unavailable.
  (d) Mark job j as assigned.

Now answer the following questions:

A. **(3 marks)** List the **sequence** of worker–job assignments the given greedy algorithm produces.

{ <A, Plumbing>, <B, Cleaning>, <C, Carpentry> }

2025CS1 1160
Krishna

B. **(3 marks)** Provide a 3×3 expertise matrix (with non-negative entries) showing a **counterexample** where the greedy algorithm fails to find the optimal plan, i.e., **maximum total expertise**.

Worker	Plumbing	Cleaning	Carpentry
A	80	10	60
B	80	70	10
C	10	70	60

ROUGH

ROUGH