Question - 17 Set- A,B,C,D

Confusion Between “Unique Elements” and “Removing Duplicates”

- For input: [1, 1, 2, 3]

- They returned: [1, 2, 3]

- But the question asked for unique elements (those appearing exactly once):

- Correct answer: [2, 3]

For this exam we accepted both, but in real algorithmic problems, these two tasks are different:

- Removing duplicates → gives set of elements

- Finding unique elements → gives elements with frequency = 1

Question - 17 Set- A,B,C,D

Using In-Built Operations Without Considering Time Complexity

Several students wrote algorithms using:

- `x in list`

- `list.count(x)`

- Repeated membership checking on lists

Operations like `in` and `count` on Python lists take O(n) time.

If placed inside a loop, the total becomes $O(n^2)$ — which the question explicitly asked you to avoid. Partial Marks for Algorithm in this case. If Time Complexity is reported as O(n), then 0.

Question - 17 Set- A,B,C,D

Incorrect Assumptions About Dictionary or Set Complexity

Some students thought:

- “Adding a key to a dictionary is O(1)”

- “Checking membership in a set is O(1)”

- Converting list to a set is O(n)

This is incorrect time analysis in worst case.

Correct facts:

Operation	Average Case	Worst Case
Dict insert/lookup	O(1)	O(n)
Set insert/lookup	O(1)	O(logn) or O(n) depending on implementation
Convert List to Set	O(nlogn)	O(nlogn)

Question - 17 Set- A,B,C,D

Missing or Incorrect Time Complexity Analysis

A surprising number of students:

- Computed time complexity as (n^2-n)/2 and said this is better than n^2 which is not the case. Both are of the same order.

Question - 17

Set- A,B,C,D

A very common mistake:

```python
for i in range(n):
    for j in range(n):
        if lst[i] != lst[j]:
            do something`
```

This does not test uniqueness.

This simply checks inequality with every other element, which is meaningless.

What could have been done was storing a boolean flag to track if a duplicate was found:

```python
for i in range(n):
    unique = True
    for j in range(n):
        if i != j and lst[i] == lst[j]:
            unique = False
            break
    if unique:
        add lst[i] to answer`
```

Even though this is $O(n^2)$, at least it is logically correct.

Many students forgot this flag and therefore produced incorrect results.

Question - 17

Set- A,B,C,D

A very common mistake:

```python
for i in range(n):
    for j in range(n):
        if lst[i] != lst[j]:
            do something`
```

This does not test uniqueness.

This simply checks inequality with every other element, which is meaningless.

What could have been done was storing a boolean flag to track if a duplicate was found:

```python
for i in range(n):
    unique = True
    for j in range(n):
        if i != j and lst[i] == lst[j]:
            unique = False
            break
    if unique:
        add lst[i] to answer`
```

Even though this is $O(n^2)$, at least it is logically correct.

Many students forgot this flag and therefore produced incorrect results.

Question - 17 Set- A,B,C,D

1. If Algorithm runs in O(nlogn) with correct Time Complexity Analysis - 10 Marks (5+5)
2. If Algorithm runs in O(n^2) with correct Time Complexity Analysis - 7.5 Marks (2.5+5)
3. If Algorithm runs in O(n^2), but students claim correct Time Complexity as O(nlogn) or O(n):- 2.5 Marks (2.5+0)

Note that the above is just a general idea and Point Adjustment has been done in many cases.

Question 15 Set A,B,C,D

Incorrect Answer (given 1.5 marks):

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

Incorrect Answer (given 1.5 marks):

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

Note: Marks have been deducted if base case is missing.
Note: Marks for even/odd case have been given only if recursive call is made **and** ‘x’ has been added/multiplied.
Note: Some marks have been deducted if result is declared and used as a global variable.

Q3 (A+C), Q1 (B+D)

In the third question of each of the subparts, the first instance at which a “new” x is formed in the code was supposed to be marked. Many of the responses simply assumed that all inputs to f would create new instances, which may be correct, is not what was asked exactly.

Merge Sort question (Q18.B.iv (A+C), Q19.B.iv (B+D))

Common errors for the merge sort implementation:

1) Unequal splitting of list.

2) Middle or last element not being included in any of the halves.

3) Used float division instead of integer division

-0.5 PENALTY FOR EACH OF THE MISTAKE

Common errors for B.iii) for the merge sort question:

1) Incorrect format without proper ordering

-1.0 penalty for the mistake

Q18.B.iv (A+C), Q19.B.iv (B+D)

- Many answers missed taking the single-element list into account.
- Many answers simply reversed the answer to the previous subpart. Please note that the invocation order is NOT the reverse of completion order.

Question 19 (Set A)

Common Errors:-
For part a, Correct assignments but incorrect sequence given
For part b, not writing the greedy and optimal assignments or scores with the counter example.

Also, for part b, write the exact matrix, the greedy and optimal assignments, with scores in the request for it to be considered.

Set A & C Question 6 (Set B & D Question 7)

Printing 100 in the initial output when the error happens before the print statement

Using nonlocal instead of global (nonlocal is for nested functions)

Output after fixing is 100, 200 (first g is printed at current value 100 then f(g) which modifies g to 200)

Defining value of g inside the function would reset it to 100 for every function call of f() but aim is to have g as a global variable

Set A & C Question 8 (Set B & D Question 5)

For the TypeError part, Common errors include skipping one or more of these type checks:

- assert type(L) == list
- assert type(L[i]) == int or type(L[i]) == float [any one of the constraints is also fine]
- assert type(b) == int and type(e) == int

There are 0.5 marks for each of them, making the total of this part to be 1.5. Answers giving vague ideas or very generic description have been given partial marks. Answers where type(b) or type(e) was allowed to be float have been given partial marks, since they can only be ints. Another common error was to plainly put a try-except block which doesn't determine the exact cause of error.

Set A & C Question 8 (Set B & D Question 5)

For the IndexError part, the most common error was to just assume that b and e can only be positive or simply giving the bound e <= len(L). These answers which do not explicitly account for negative indices have been given 1/1.5 marks.

Another common error was that many answers just checked for b<=e or e-b <= len(L). Both of these answers do not prevent indexErrors for all scenarios (e.g. what if e = len(L)*1.5 and b = len(L)). All such answers were given 0/1.5.

Another error was that some answers just checked for b <= len(L). These have been given partial marks, since e could be greater than len(L) and cause error.