

Pset 3 outputs

Q1

The screenshot shows a code editor window titled "prob1_temperaturealert.py". The code defines a function `preprocess` that takes a list of temperatures and returns the next alert and its prefix. It then processes queries like "NEXT" and "COUNT" on the resulting list. The terminal output shows the execution of the script and the results of the queries.

```
def preprocess(alertDays):
    return nextAlert, prefix
# INPUT
temp = [73, 74, 75, 71, 69, 72, 76, 73]
K = 3
queries = [
    ("NEXT", 2),
    ("COUNT", 1, 5),
    ("NEXT", 6)
]
# PROCESS
alertDays = build_alert_days(temp, K)
nextAlert, prefix = preprocess(alertDays)
print("Alert Days:", alertDays)
for q in queries:
    if q[0] == "NEXT":
        x = q[1]
        print("NEXT", x, "->", nextAlert[x])
    else:
        l, r = q[1], q[2]
        print("COUNT", l, r, "->", prefix[r + 1] - prefix[l])
```

TERMINAL

```
PS C:\anas_3rd_sem\logic forge\PSET3> & C:/Python34/python.exe "c:/anas_3rd_sem/logic forge/PSET3/prob1_temperaturealert.py"
PS C:\anas_3rd_sem\logic forge\PSET3> & C:/Python34/python.exe "c:/anas_3rd_sem/logic forge/PSET3/prob1_temperaturealert.py"
Alert Days: [4, 3, 3, 6, 5, 6, 7, 0]
NEXT 2 -> 3
COUNT 1 5 -> 5
NEXT 6 -> 7
PS C:\anas_3rd_sem\logic forge\PSET3>
```

Q2

The screenshot shows a code editor window with two tabs: "prob1_temperaturealert.py" and "prob2_emergencyevac.py". The "prob2_emergencyevac.py" tab is active, displaying a binary search algorithm to find the minimum number of boats required to transport people across a river. The terminal output shows the execution of the script and the result.

```
def min_boats(weights, limit):
    weights.sort()
    left = 0
    right = len(weights) - 1
    boats = 0
    while left <= right:
        if weights[left] + weights[right] <= limit:
            left += 1 # light person used
            right -= 1 # heavy person always used
            boats += 1
    return boats
weights = [3, 2, 2, 1]
limit = 3
print("Minimum boats needed:", min_boats(weights, limit))
```

TERMINAL

```
PS C:\anas_3rd_sem\logic forge\PSET3> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic forge\PSET3/prob2_emergencyevac.py"
Minimum boats needed: 3
PS C:\anas_3rd_sem\logic forge\PSET3>
```

Q3

```
prob3.broadcastnetwork.py > ...
1  def broadcast_network_feed(N, Q, K, operations):
2
3      for mid, t, sender, critical in reversed(messages):
4          if sender == u or sender in subscriptions[u]:
5              if mid in user_messages[sender]:
6                  feed.append((t, critical, mid))
7
8          if len(feed) == 10:
9              break
10
11      if not feed:
12          print("EMPTY")
13      else:
14          feed.sort(key=lambda x: (-x[0], -x[1]))
15          print(" ".join(str(x[2]) for x in feed))
16
17
18  N, Q, K = 3,9,2
19  operations = [
20      "S 1 2",
21      "S 1 3",
22      "B 2 5",
23      "B 3 9",
24      "B 3 9",
25      "B 3 9"
26  ]
27
28
29
30
31
32
33
34
35
36
37
38
39
```

F 1
F 2
2 1
3 2
1

PS C:\anas_3rd_sem\logic forge\PSET3> & C:/Python314/python.exe "c:/anas_3rd_sem/logic forge/PSET3/prob3_broadcastnetwork.py"

2 1
3 2
1

Q4

```
prob4_scrambledkeywoard.py > ⌂ find_anagrams
1  def find_anagrams(s, p):
2
3      freq_window[ord(s[i - window_size]) - ord('a')] -= 1
4
5      if freq_window == freq_p:
6          result.append(i - window_size + 1)
7
8  return result
9
10
11 def main():
12     s = "cbaebabacd"
13     p = "abc"
14
15     ans = find_anagrams(s, p)
16     print(ans)
17
18
19 if __name__ == "__main__":
20     main()
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

2 1
3 2
1

PS C:\anas_3rd_sem\logic forge\PSET3> & C:/Python314/python.exe "c:/anas_3rd_sem/logic forge/PSET3/prob3_broadcastnetwork.py"

2 1
3 2
1

Traceback (most recent call last):
 File "c:/anas_3rd_sem/logic forge\PSET3\prob3_broadcastnetwork.py", line 70, in <module>
 operations.append(input().strip())
 ~~~~~^M
KeyboardInterrupt
PS C:\anas\_3rd\_sem\logic forge\PSET3> & C:/Python314/python.exe "c:/anas\_3rd\_sem/logic forge/PSET3/prob4\_scrambledkeywoard.py"
[0, 6]
PS C:\anas\_3rd\_sem\logic forge\PSET3>

## Q5

The screenshot shows the VS Code interface with the file `prob5_longestmirror.py` open. The code implements a dynamic programming solution for finding the longest mirrored phrase in a string. It uses a 2D DP array where `dp[i][j]` represents the length of the longest mirrored phrase from index `i` to `j`. The code iterates through all possible lengths and indices, updating the DP table based on matching characters at the current index and its mirror index.

```
prob5_longestmirror.py
```

```
def longest_mirrored_phrase(s):
    n = len(s)
    dp = [[0] * n for _ in range(n)]
    # single characters
    for i in range(n):
        dp[i][i] = 1
    # build table bottom-up
    for length in range(2, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            if s[i] == s[j]:
                if length == 2:
                    dp[i][j] = 2
                else:
                    dp[i][j] = 2 + dp[i + 1][j - 1]
            else:
```

TERMINAL

```
PS C:\anas_3rd_sem\logic_forge\PSET3> & C:/Python314/python.exe "c:/anas_3rd_sem/logic_forge/PSET3/prob5_longestmirror.py"
4
PS C:\anas_3rd_sem\logic_forge\PSET3>
```

## Q6

The screenshot shows the VS Code interface with the file `prob6_suspicious_device.py` open. The code defines a function `find_suspicious_device` that takes a list of integers and returns the integer that appears most frequently. It uses a dictionary `freq` to count the occurrences of each number. The main function `main` prints the result of calling `find_suspicious_device` with a hardcoded input list. The script is run from the terminal, and it outputs the value 5, which is the most frequent number in the input list [2, 1, 2, 5, 3, 2].

```
prob6_suspicious_device.py
```

```
def find_suspicious_device(nums):
    freq = {}
    n = len(nums) // 2
    for x in nums:
        freq[x] = freq.get(x, 0) + 1
        if freq[x] == n:
            return x
def main():
    # HARD-CODED INPUT
    nums = [2, 1, 2, 5, 3, 2]
    print(find_suspicious_device(nums))
if __name__ == "__main__":
    main()
```

TERMINAL

```
PS C:\anas_3rd_sem\logic_forge\PSET3> ./anas_3rd_sem/logic_forge/PSET3/prob5_longestmirror.py
4
PS C:\anas_3rd_sem\logic_forge\PSET3> ^C
PS C:\anas_3rd_sem\logic_forge\PSET3> & C:/Python314/python.exe "c:/anas_3rd_sem/logic_forge/PSET3/prob6_suspicious_device.py"
2
PS C:\anas_3rd_sem\logic_forge\PSET3>
```