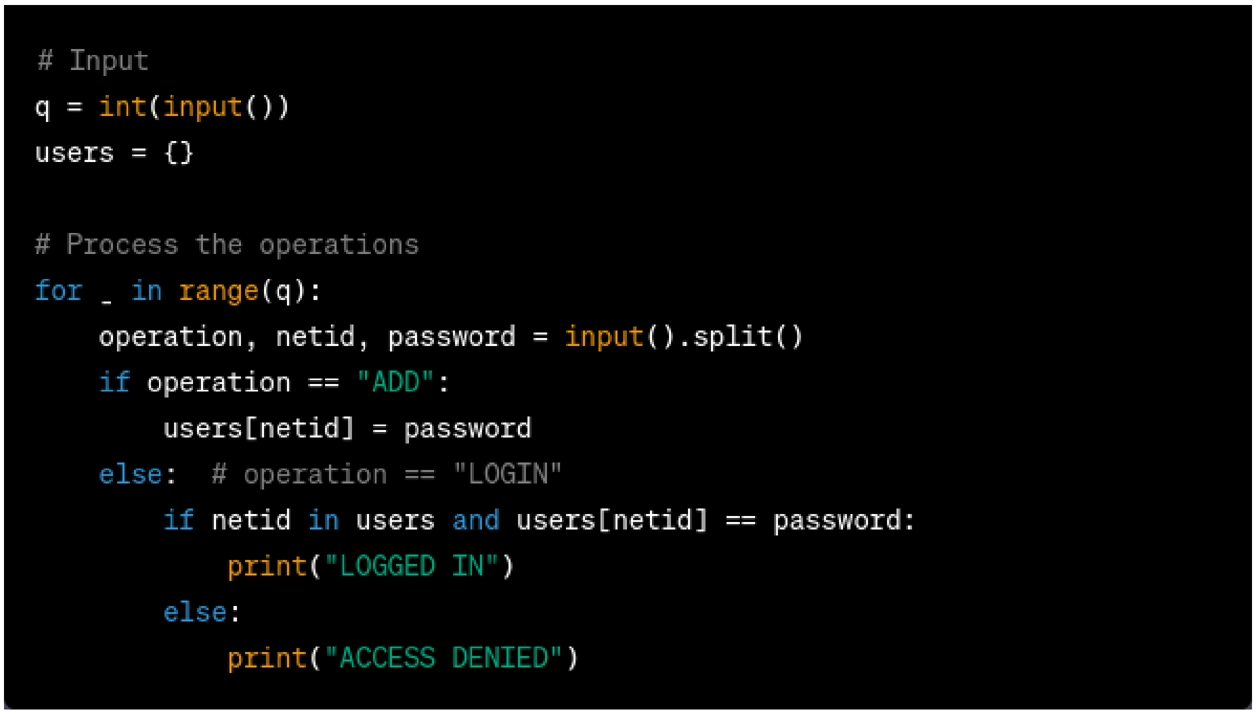
Fall 23 Div II Week 4 - Solution Sketches

(taken from editorials of original problems)

Problem A

(Problem by Pedro Paredes)

To create a new netid system, we can use a dictionary/symbol table/map data structure to store user information where the netid is the key and the password is the associated value. We initialize an empty dictionary and then process the given operations. For "ADD" operations, we add the user's netid and password to the dictionary. For "LOGIN" operations, we check if the netid exists in the dictionary and if the stored password matches the provided password. If it does, we print "LOGGED IN," otherwise, we print "ACCESS DENIED." Here's the Python code for this solution:



```python
# Input
q = int(input())
users = {}

# Process the operations
for _ in range(q):
    operation, netid, password = input().split()
    if operation == "ADD":
        users[netid] = password
    else:  # operation == "LOGIN"
        if netid in users and users[netid] == password:
            print("LOGGED IN")
        else:
            print("ACCESS DENIED")
```

(bonus question: do you recognize the above color scheme

Problem B

(Source: Codeforces Round 817 (Div. 4) Problem C)

For each test case, we can solve this problem by creating a dictionary/symbol table/map to count the occurrences of each word written by the players. We initialize a dictionary for each test case and process the words written by the three players, incrementing the count for each word. After processing all the words, we calculate the points earned by each player based on the word counts. The player who wrote a word alone (count = 1) gets 3 points, players who wrote a word together (count = 2) get 1 point each, and if a word was written by all (count = 3), nobody gets any points.

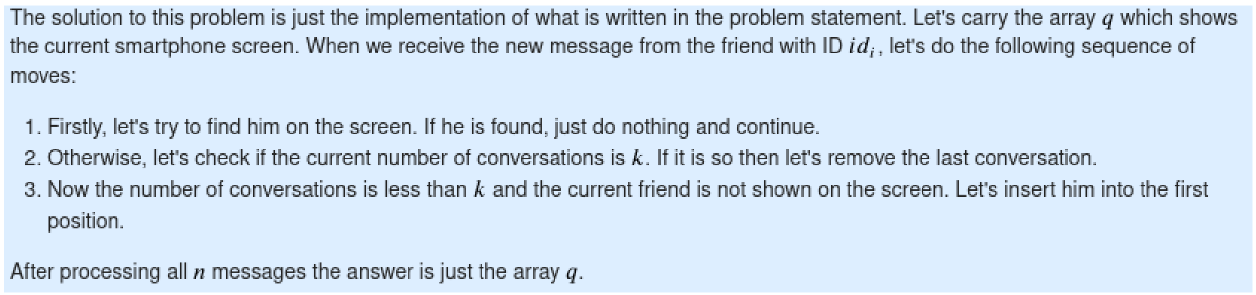
Problem C

(Source: Codeforces Round #797 (Div. 3) Problem C)
To solve it you have to simulate this process, by keeping track of the current time (set a variable curTime = 0):

- Take a task from the queue
- Take as time the maximum from the current time and from the arrival time of the task (curTime = max(curTime, s), where s is the arrival time)
- Subtract the current time from the time when the task was done (d = f - curTime) where f is the time when the task was done and d is the duration we are trying to find)
- Replace the current time with the time the task was done (curTime = d)
- If there is a task in the queue, go to item 1

Problem D

(Source: Codeforces Round #590 (Div. 3) Problem B1)


The solution to this problem is just the implementation of what is written in the problem statement. Let's carry the array q which shows the current smartphone screen. When we receive the new message from the friend with ID id_i, let's do the following sequence of moves:

1. Firstly, let's try to find him on the screen. If he is found, just do nothing and continue.
2. Otherwise, let's check if the current number of conversations is k. If it is so then let's remove the last conversation.
3. Now the number of conversations is less than k and the current friend is not shown on the screen. Let's insert him into the first position.

After processing all n messages the answer is just the array q.

Problem E

(source: Codeforces Round 179 (Div. 1) Problem A)
You need to use the concept of *difference arrays*. Make sure you know how it works first (here is an article about it). If the problem were about applying one of the operations, then the solution would be simply to use difference arrays: for each interval [l, r] add +d to the difference array at position l and a -d at r+1. Then iterate through the array and keep track of the cumulative sum.

To solve the problem we now use difference arrays on each of the k queries: for each interval of operations [x, y], add +1 to a difference array at position x and a -d at r+1. This difference array will represent how many times each operation will be applied. Now create a second difference array for the original array a. Go through each operation and keep track of the cumulative sum given by the operations difference array. If the sum is ct for operation [l,r,d] (meaning you have to apply that operation ct times), then add +d*ct to position l and -d*ct to position r+1.