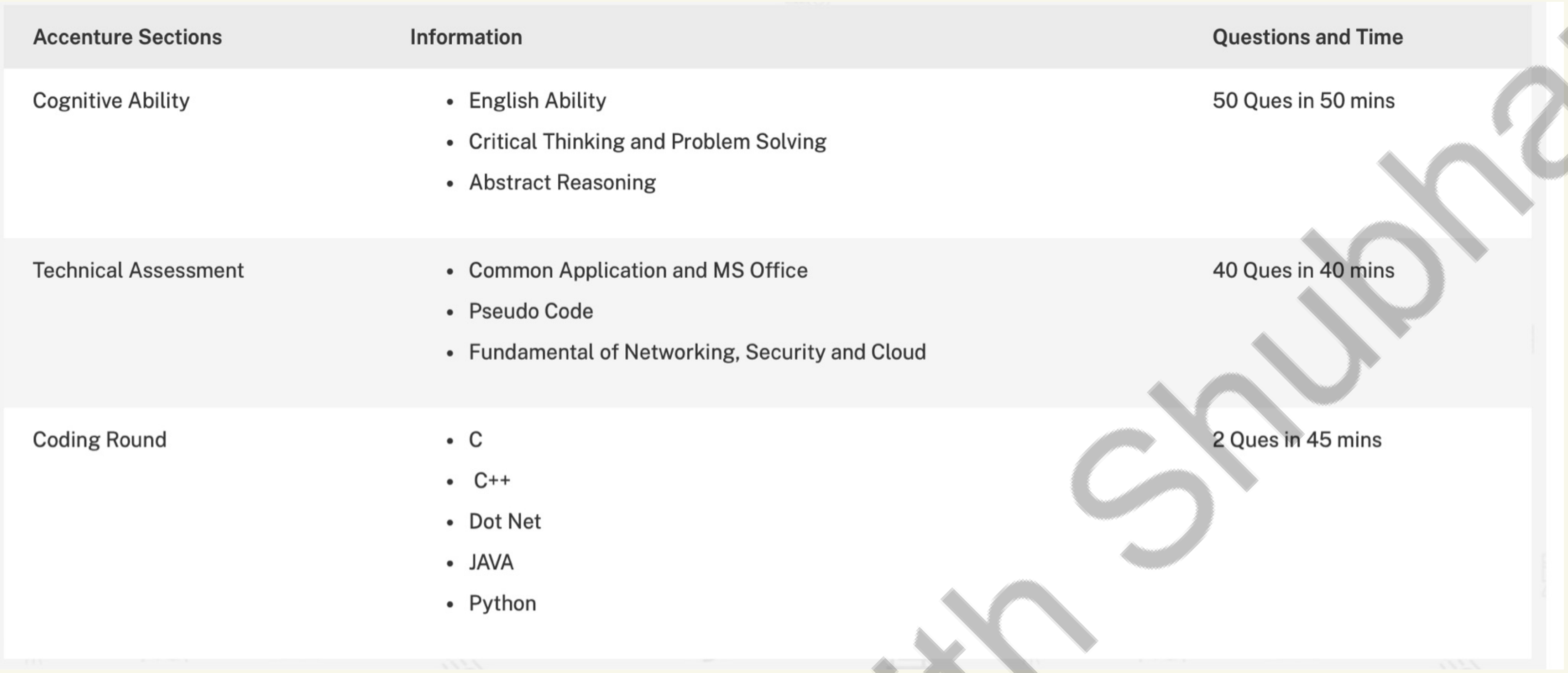


Accenture Sections	Information	Questions and Time
Cognitive Ability	• English Ability • Critical Thinking and Problem Solving • Abstract Reasoning	50 Ques in 50 mins
Technical Assessment	• Common Application and MS Office • Pseudo Code • Fundamental of Networking, Security and Cloud	40 Ques in 40 mins
Coding Round	• C • C++ • Dot Net • JAVA • Python	2 Ques in 45 mins

DEBUG WITH SHUBHAM

Accenture Technical Assessment Detailed Overview

14-SEP-2024 Coding Question

https://www.youtube.com/@DebugWithShubham

https://www.linkedin.com/in/debugwithshubham/

https://www.instagram.com/debugwithshubham/

https://topmate.io/debugwithshubham

https://t.me/debugwithshubham

Question -1

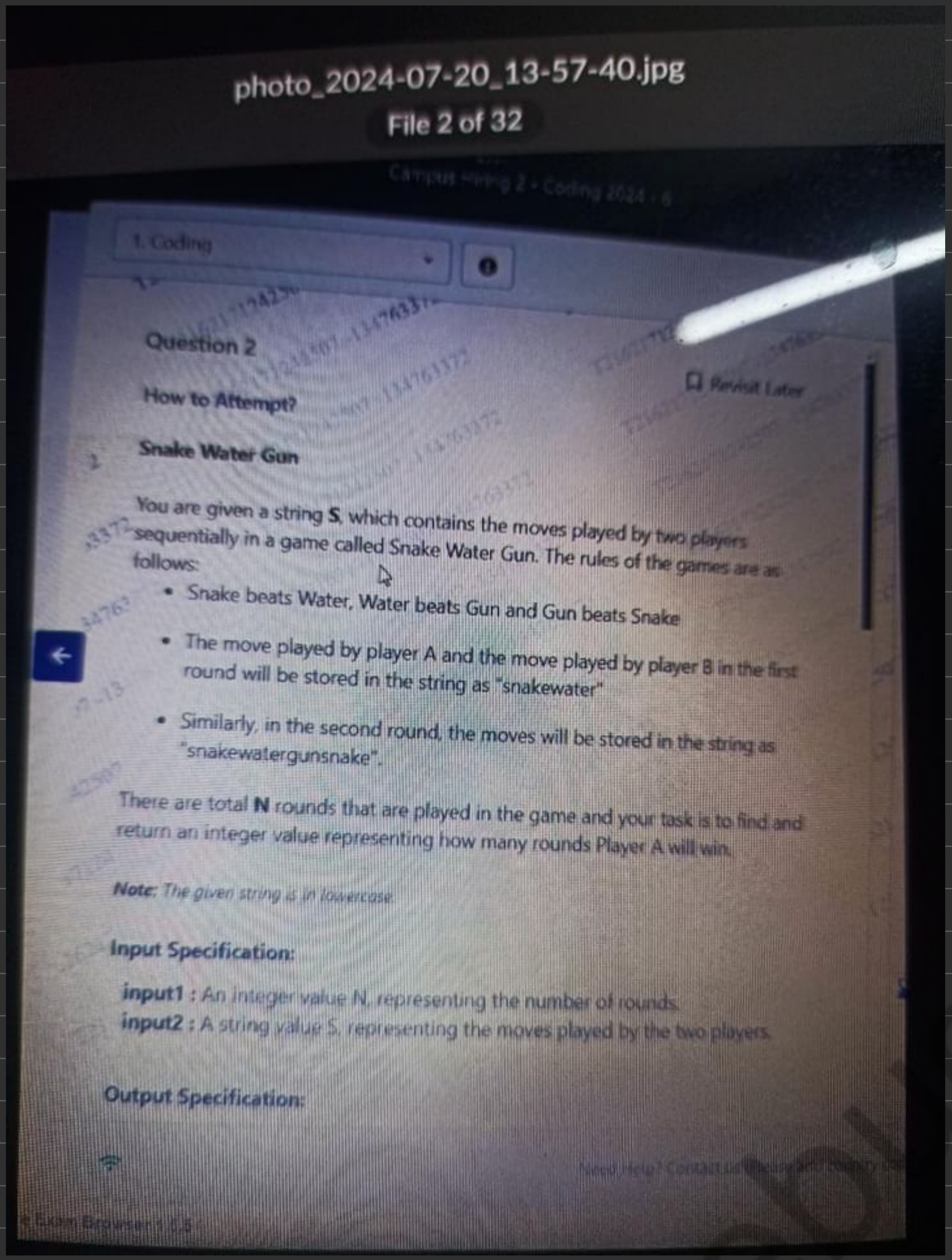


photo_2024-07-20_13-57-40.jpg

File 2 of 32

Campus Hiring 2 - Coding 2024 - 6

Question 2

How to Attempt?

Snake Water Gun

You are given a string **S**, which contains the moves played by two players sequentially in a game called Snake Water Gun. The rules of the games are as follows:

- Snake beats Water, Water beats Gun and Gun beats Snake
- The move played by player A and the move played by player B in the first round will be stored in the string as "snakewater"
- Similarly, in the second round, the moves will be stored in the string as "snakewatergunsnake".

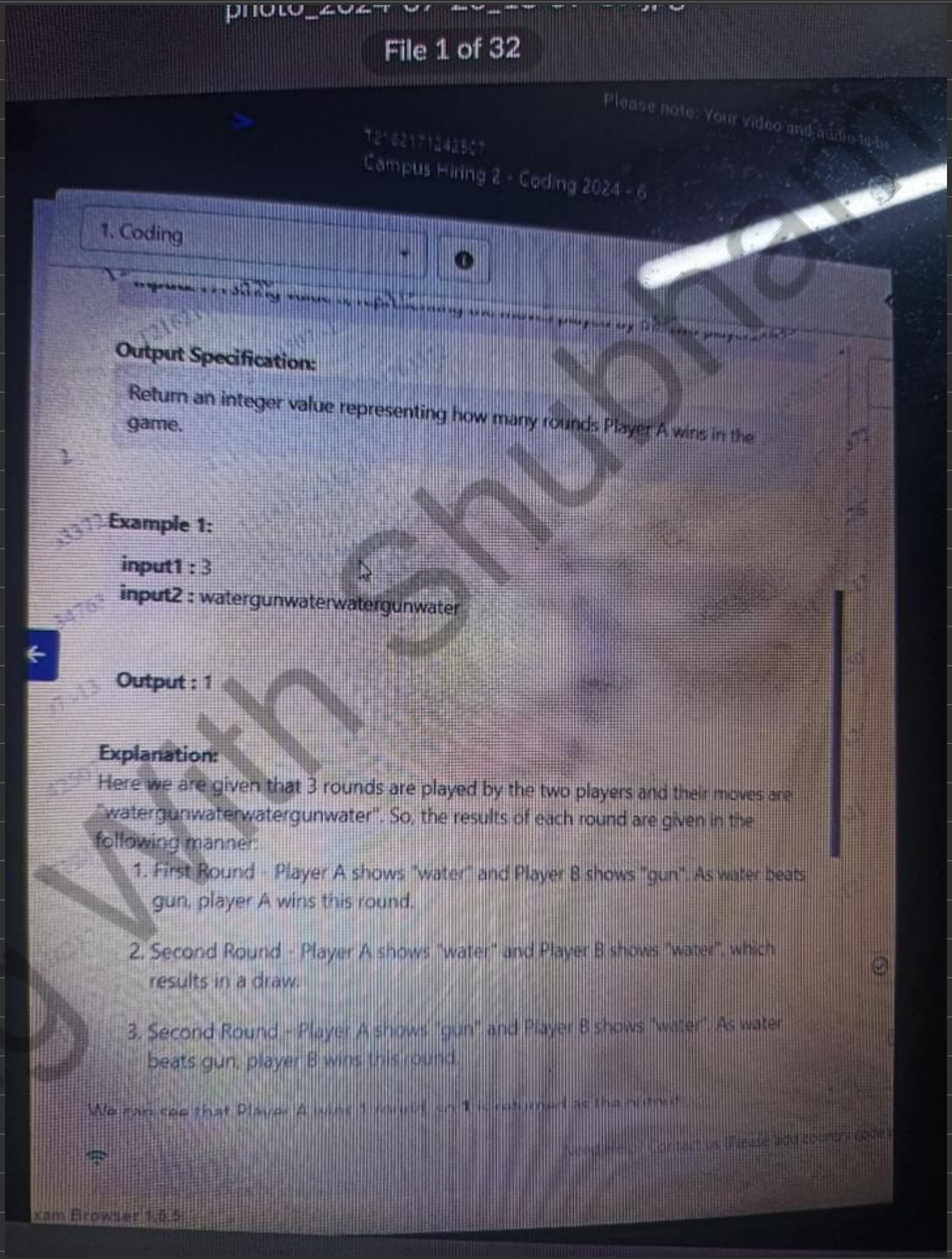
There are total **N** rounds that are played in the game and your task is to find and return an integer value representing how many rounds Player A will win.

Note: *The given string is in lowercase*

Input Specification:

input1 : An integer value N, representing the number of rounds.
input2 : A string value S, representing the moves played by the two players.

Output Specification:



File 1 of 32

Please note: Your video and audio is be

Campus Hiring 2 - Coding 2024 - 6

Output Specification:

Return an integer value representing how many rounds Player A wins in the game.

Example 1:

input1 : 3
input2 : watergunwaterwatergunwater

Output : 1

Explanation:
Here we are given that 3 rounds are played by the two players and their moves are "watergunwaterwatergunwater". So, the results of each round are given in the following manner:

1. First Round - Player A shows "water" and Player B shows "gun". As water beats gun, player A wins this round.
2. Second Round - Player A shows "water" and Player B shows "water", which results in a draw.
3. Second Round - Player A shows "gun" and Player B shows "water". As water beats gun, player B wins this round.

We can see that Player A wins 1 round, so 1 is returned as the output.

C++

```cpp
#include <iostream>
#include <vector>
#include <string>

int main() {
    std::string s = "watergunwaterwatergunwater";
    std::vector<std::string> delimiters = {"water", "gun", "snake"};
    std::vector<std::string> ans;
    int cnt = 0;
    while (!s.empty()) {
        for (const std::string& move : delimiters) {
            if (s.find(move) == 0) {
                ans.push_back(move);
                s = s.substr(move.length());
                break;
            }
        }
    }

    int n = ans.size();
    for (int i = 0; i < n; i += 2) {
        std::string playerA = ans[i];
        std::string playerB = ans[i + 1];

        if (playerA == "snake" && playerB == "water") {
            cnt++;
        } else if (playerA == "water" && playerB == "gun") {
            cnt++;
        } else if (playerA == "gun" && playerB == "snake") {
            cnt++;
        }
    }
    std::cout << cnt << std::endl;

    return 0;
}
```

Python

```python
s = "watergunwaterwatergunwater"
print(cnter','gun','snake']
cnt = 0
ans = []
while s:
    for i in d:
        if s.startswith(i):
            ans.append(i)
            s = s[len(i):]
            break
n = len(ans)
for i in range(0,n,2):
    if ans[i] == 'snake' and ans[i+1] == 'water':
        cnt +=1
    elif  ans[i] == 'water' and ans[i+1] == 'gun':
        cnt +=1
    elif  ans[i] == 'gun' and ans[i+1] == 'snake':
        cnt +=1
print(cnt)
```

Java

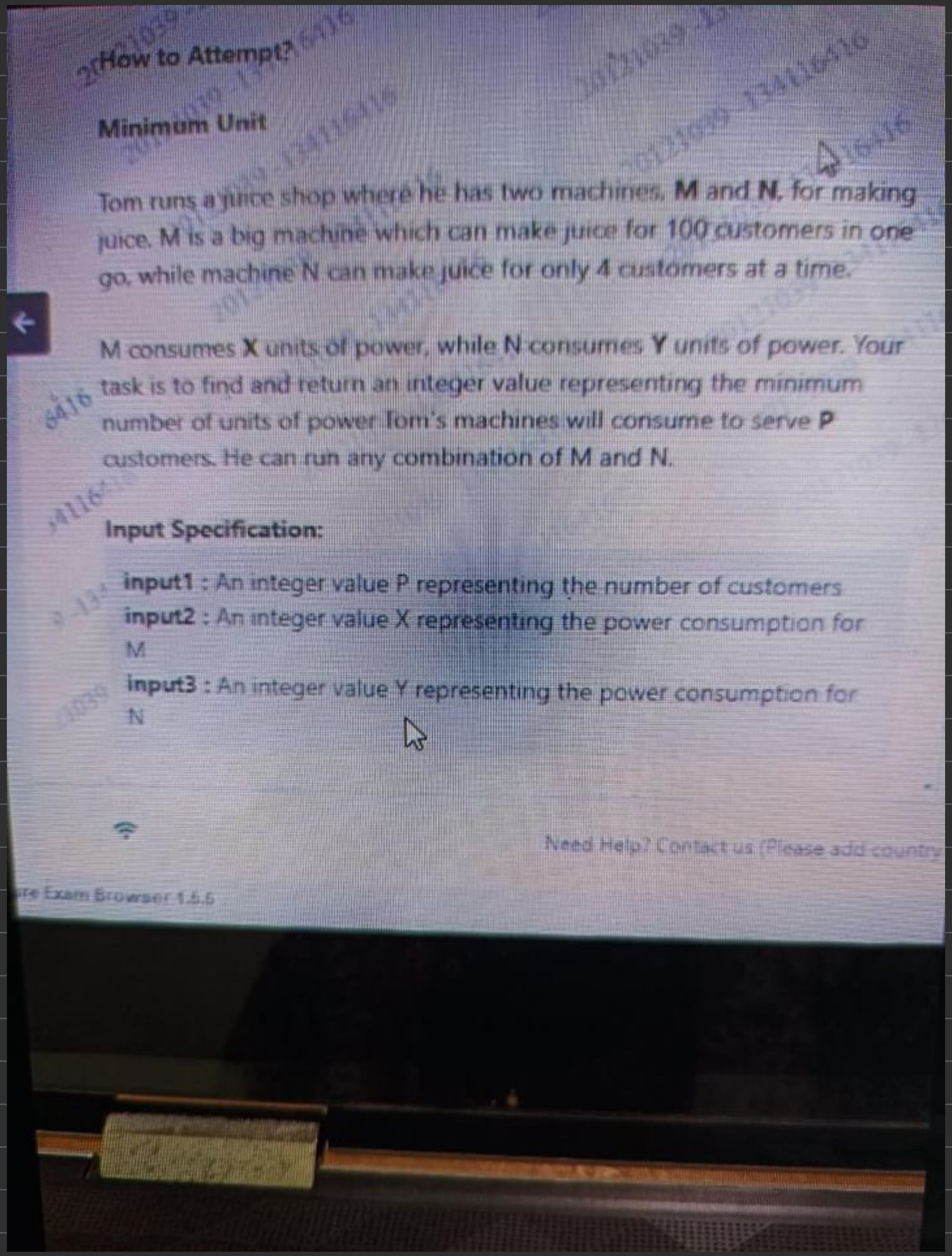
```java
import java.util.ArrayList;
import java.util.List;

public class SnakeWaterGun {
    public static void main(String[] args) {
        String s = "watergunwaterwatergunwater";
        String[] d = {"water", "gun", "snake"};
        int cnt = 0;
        List<String> ans = new ArrayList<>();
        while (!s.isEmpty()) {
            for (String move : d) {
                if (s.startsWith(move)) {
                    ans.add(move);
                    s = s.substring(move.length());
                    break;
                }
            }
        }

        int n = ans.size();
        for (int i = 0; i < n; i += 2) {
            String playerA = ans.get(i);
            String playerB = ans.get(i + 1);

            if (playerA.equals("snake") && playerB.equals("water")) {
                cnt++;
            } else if (playerA.equals("water") && playerB.equals("gun")) {
                cnt++;
            } else if (playerA.equals("gun") && playerB.equals("snake")) {
                cnt++;
            }
        }
        System.out.println(cnt);
    }
}
```

Question-2



How to Attempt?

Minimum Unit

Tom runs a juice shop where he has two machines, **M** and **N**, for making juice. M is a big machine which can make juice for 100 customers in one go, while machine N can make juice for only 4 customers at a time.

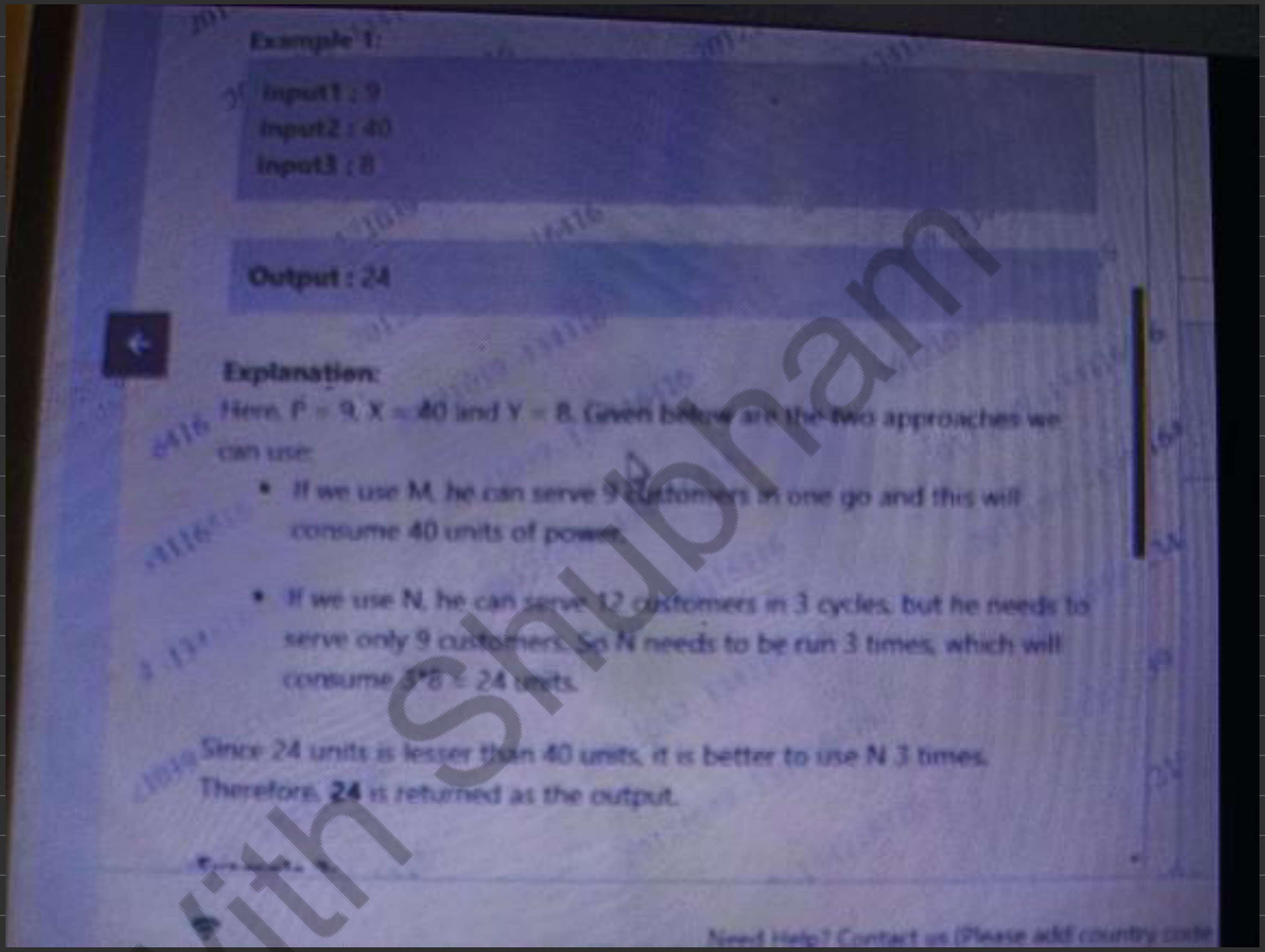
M consumes **X** units of power, while N consumes **Y** units of power. Your task is to find and return an integer value representing the minimum number of units of power Tom's machines will consume to serve **P** customers. He can run any combination of M and N.

Input Specification:

input1 : An integer value P representing the number of customers
input2 : An integer value X representing the power consumption for M
input3 : An integer value Y representing the power consumption for N

Need Help? Contact us (Please add country

re Exam Browser 1.5.5



Example 1:

Input1 : 9
Input2 : 40
Input3 : 8

Output : 24

Explanation:
Here, P = 9, X = 40 and Y = 8. Given below are the two approaches we can use:

- If we use M, he can serve 9 customers in one go and this will consume 40 units of power.
- If we use N, he can serve 12 customers in 3 cycles, but he needs to serve only 9 customers. So N needs to be run 3 times, which will consume 3*8 = 24 units.

Since 24 units is lesser than 40 units, it is better to use N 3 times. Therefore, **24** is returned as the output.

Example 2:

Need Help? Contact us (Please add country code

Python

```python
import math
def minimum_power_greedy(P, X, Y):
    use_big_machines = P // 100
    remaining_customers = P % 100
    power_M = use_big_machines * X
    power_M_remaining = X  #40
    power_N_remaining = math.ceil(remaining_customers / 4) * Y # 24
    total_power_use_big_machine = power_M + power_M_remaining
    total_power_use_small_machine = power_M + power_N_remaining
    return min(total_power_use_big_machine, total_power_use_small_machine)
P = 9
X = 40
Y = 8
print(minimum_power_greedy(P, X, Y))
```

C++

```cpp
#include <iostream>
#include <cmath> // For ceil()

using namespace std;

int minimumPowerGreedy(int P, int X, int Y) {
    int useBigMachines = P / 100;
    int remainingCustomers = P % 100;
    int powerM = useBigMachines * X;
    int powerMRemaining = X;
    int powerNRemaining = ceil(remainingCustomers / 4.0) * Y;

    int totalPowerUseBigMachine = powerM + powerMRemaining;
    int totalPowerUseSmallMachine = powerM + powerNRemaining;

    return min(totalPowerUseBigMachine, totalPowerUseSmallMachine);
}

int main() {
    int P = 9;
    int X = 40;
    int Y = 8;
    cout << minimumPowerGreedy(P, X, Y) << endl;
    return 0;
}
```

Java

```java
import java.lang.Math;

public class MinimumPower {

    public static int minimumPowerGreedy(int P, int X, int Y) {
        int useBigMachines = P / 100;
        int remainingCustomers = P % 100;

        int powerM = useBigMachines * X;
        int powerMRemaining = X;
        int powerNRemaining = (int) Math.ceil(remainingCustomers / 4.0) * Y;

        int totalPowerUseBigMachine = powerM + powerMRemaining;
        int totalPowerUseSmallMachine = powerM + powerNRemaining;

        return Math.min(totalPowerUseBigMachine,
totalPowerUseSmallMachine);
    }

    public static void main(String[] args) {
        int P = 9;
        int X = 40;
        int Y = 8;
        System.out.println(minimumPowerGreedy(P, X, Y));
    }
}
```