Problem 1

```python
def identify_baby_boss_habit(input_string: str) -> str:
    n = len(input_string)
    count_S = 0
    i = 0
    while i < n:
        if input_string[i] == "R":
            if count_S == 0:
                return "Bad boy"
            else:
                while i < n and input_string[i] == "R":
                    count_S = max(count_S - 1, 0)
                    i += 1
        elif input_string[i] == "S":
            count_S += 1
            i += 1

    return "Good boy" if count_S == 0 else "Bad boy"
```

From the question, it can be concluded that a boss baby will be classified as a bad boy based on the following criteria:

- While looping through the input_string, if ‘R’ is found and
    - he has already sought revenge for every previous shot
    - or he has not taken a single shot.
- If he does not have a chance to seek revenge for every single shot.

Script Explanation

```
while i < n:
    if input_string[i] == "R":
        if count_S == 0:
            return "Bad boy"
```

count_S tells how many shots he has taken from the kid and does not have any chance to shoot back. Therefore, in the script, if 'R' is found but count_S = 0, this means he is a bad boy because he shoots first.

```
else:
    while i < n and input_string[i] == "R":
        count_S = max(count_S - 1, 0)
        i += 1
```

If count_S>0 while 'R' is found, I will subtract 1 from count_S because the baby boss has taken revenge . I use the while loop because the baby can shoot back more than 1 time if he needed. This means that I need to keep count_S >=0 because the numbers of shot taken should always be >=0.

```
elif input_string[i] == "S":
    count_S += 1
    i += 1
```

If "S" is found, count_S will be increased by 1 because the baby boss has been shot.

```
return "Good boy" if count_S == 0 else "Bad boy"
```

After looping through the input, if he does not have a chance to seek revenge for every single shot, he will be classified as a bad boy. Otherwise, he will be classified as a good boy.

Complexity Analysis

```
# Time Complexity O(n)
# Space Complexity O(n)
```

Time Complexity : O (n) => loop through the string of length n

Space Complexity : O(n) => the input is the string of length n