# TROLL CODE

Integrantes:

- Roberth Gancino
- Anthony Goyes
- Danny Iñaguazo

GitHub: https://github.com/NwDann/Taller01-MN.git

Nickname en CSACADEMY: "**Reik Polamina**"

Realmente costó moldear el código para este ejercicio debido a algunos problemas presentes para varias cadenas de bits. Entonces el primer resultado fue una mitad de nota:

Source Summary **Results** Compilation messages

Score: 55.88/100 (33 ms - 4268 KB)

| Test Number | CPU Usage | Memory Usage | Result |
|---|---|---|---|
| 102 | 24 ms | 4248 KB | Invalid operation! Expected 'A' or 'Q' |
| 101 | 24 ms | 4184 KB | Ok! Used 5 queries. (n+0) |
| 100 | 27 ms | 4260 KB | Ok! Used 87 queries. (n-1) |
| 99 | 27 ms | 4256 KB | Ok! Used 82 queries. (n-1) |
| 98 | 26 ms | 4260 KB | Ok! Used 65 queries. (n-7) |
| 97 | 27 ms | 4252 KB | Ok! Used 86 queries. (n-1) |
| 96 | 28 ms | 4268 KB | Invalid operation! Expected 'A' or 'Q' |
| 95 | 26 ms | 4264 KB | Ok! Used 71 queries. (n+0) |
| 94 | 26 ms | 4256 KB | Ok! Used 97 queries. (n-1) |
| 93 | 26 ms | 4256 KB | Invalid operation! Expected 'A' or 'Q' |

*Ilustración 1: Puntuación de la resolución inicial.*

Los casos que nos originaban errores fueron manejados correctamente mediante condiciones y consultas al "Troll". El resultado fue excelente:

Input Output Stderr Compilation Execution Examples **Submission**

Score: 100/100 (29 ms - 4008 KB)

| Test Number | CPU Usage | Memory Usage | Result |
|---|---|---|---|
| 102 | 24 ms | 3992 KB | Ok! Used 6 queries. (n+1) |
| 101 | 24 ms | 3988 KB | Ok! Used 6 queries. (n+1) |
| 100 | 26 ms | 4000 KB | Ok! Used 88 queries. (n+0) |
| 99 | 26 ms | 4000 KB | Ok! Used 84 queries. (n+1) |
| 98 | 25 ms | 3996 KB | Ok! Used 66 queries. (n-6) |
| 97 | 27 ms | 4000 KB | Ok! Used 88 queries. (n+1) |
| 96 | 28 ms | 3996 KB | Ok! Used 80 queries. (n+1) |
| 95 | 24 ms | 3996 KB | Ok! Used 71 queries. (n+0) |

Execution Details | Compile | Run input | Run examples | Submit

*Ilustración 2: Puntuación de la resolución mejorada.*

Finalmente, los siguientes códigos fueron los postulados como soluciones. En el segundo programa, se señalan las correcciones utilizadas para mejorar el algoritmo y robustecer el manejo de cadenas de bits.

**CODIGO INICIAL (Score: 55.8/100, Fecha: 26/06/2024, Hora: 9:30)**

```
def guessing():
    global sequence, prev_correctBits
    for i in range(N):
        sequence[i] = 1 - sequence[i]  # Changing from 0 to 1 and from 1
to 0
        outPut = "Q " + " ".join(map(str, sequence))
        print(outPut)
        sys.stdout.flush()
        correctBits = int(input())

        if correctBits == N:
            return True  # Correct sequence

        elif correctBits < N and correctBits > prev_correctBits:
            prev_correctBits = correctBits

        else:
            sequence[i] = 1 - sequence[i]  # Reversing the change

    return False

import sys

N = int(input())
sequence = [0] * N
prev_correctBits = 0

flag = guessing()

if flag:
    outPut = "A " + " ".join(map(str, sequence))
    print(outPut)
    sys.stdout.flush()
else:
    print("Sequence not guessed")
```

**CODIGO MEJORADO (Score: 100/100, Fecha: 26/06/2024, Hora: 16:31)**

El código en rojo es el añadido, permitiendo ahorrar consultas en ciertos casos con la primera consulta dentro de la función “guessing”. Después comprobamos si nos conviene trabajar con una secuencia de bits con la mayoría de ceros o unos.

```
def guessing(N):
    global sequence
    outPut = "Q " + " ".join(map(str, sequence))
    print(outPut)
    prev_correctBits = int(input())
    sys.stdout.flush()

    if prev_correctBits < N/2:
        sequence = [1] * N
        prev_correctBits = N - prev_correctBits

    for i in range(N):
        sequence[i] = 1 - sequence[i]  # Changing from 0 to 1 and from 1
to 0
        outPut = "Q " + " ".join(map(str, sequence))
        print(outPut)
        correctBits = int(input())
        sys.stdout.flush()

        if correctBits == N:
            return True  # Correct sequence

        elif correctBits < N and correctBits > prev_correctBits:
            prev_correctBits = correctBits

        else:
            sequence[i] = 1 - sequence[i]  # Reversing the change
            if (i == N - 1) and (correctBits == N - 1):
                return True

    return False

# ------------------------------MAIN-------------------------------------
--

import sys

N = int(input())
sequence = [0] * N

flag = guessing(N)
```

```
if flag:
    outPut = "A " + " ".join(map(str, sequence))
    print(outPut)
    sys.stdout.flush()
else:
    print("Sequence not guessed")
```