

# Duplicated Binary Strings

Carlos spends his summer holiday studying **duplicated binary strings**. A duplicated binary string is a non-empty string  $T$  such that:

- $T$  contains only the characters 0 and 1 (that is,  $T$  is a **binary string**).
- $T$  can be written in the form  $T = \overline{UU}$ , where  $U$  is an arbitrary binary string and the operation  $\overline{ab}$  denotes the concatenation of the strings  $a$  and  $b$  (i.e., writing them one after the other as a single string).

For example, 0000 and 011011 are duplicated binary strings, but 01, 0110, and 000 are not.

Define the **strength** of a binary string  $S$  as the number of **distinct** contiguous duplicated substrings present in  $S$ . Two substrings are considered different if they differ in at least one character.

This problem consists of two parts, with each subtask associated with either **Part I** or **Part II**. You may solve the subtasks in any order; in particular, you are not required to complete all of Part I before attempting Part II.

## Part I

Carlos sends you a binary string  $S$ , and your task is to calculate its strength.

### Implementation Details

You should implement the following procedure.

```
int count_duplicated(std::string S)
```

- $S$ : binary string of length  $N$
- This procedure is called exactly once for each test case.

The procedure should return an integer  $K$ , the number of distinct contiguous duplicated substrings present in  $S$ .

## Constraints

- $4 \leq N \leq 100$
- $S[i]$  is either 0 or 1 for each  $i$  such that  $0 \leq i < N$ .

## Subtasks

Subtask	Score	Additional Constraints
1	6	$N = 4$
2	9	No additional constraints.

## Examples

### Example 1

Consider the following call.

```
count_duplicated("0101")
```

There is only one duplicated binary substring in  $S$ , which is 0101. Therefore the procedure should return 1.

### Example 2

Consider the following call.

```
count_duplicated("0000")
```

There are two duplicated binary substrings in  $S$ : 00 and 0000. Hence, the procedure should return 2.

Note that although the substring 00 appears three times in  $S$ , it is counted only once in the final answer.

## Part II

Carlos wonders what the minimum and maximum strength of a binary string  $S$  can be.

Your task is to construct binary strings **of length** 100 that contain as few or as many duplicated substrings as possible. You will receive a score based on the number of duplicated substrings.

## Subtasks

This part consists of 2 **output-only** subtasks with **partial scoring**.

Subtask	Score	Constraint
3	25	Minimize the strength of $S$ .
4	60	Maximize the strength of $S$ .

## Implementation Details

For each subtask, you should either

- submit an output file consisting of a binary string of length 100, or
- return a binary string in your program to a grader procedure call.

Each output file must be in the following format:

```
S
```

To construct the required binary strings in your solution program, you should implement the following procedures.

```
std::string find_weakest()
```

- If an output file for Subtask 3 is provided in your submission, this procedure will not be called.
- Otherwise, the procedure is called in Subtask 3 exactly once.

The procedure should return a binary string  $S$  of length  $N = 100$  with minimum strength.

```
std::string find_strongest()
```

- If an output file for Subtask 4 is provided in your submission, this procedure will not be called.
- Otherwise, the procedure is called in Subtask 4 exactly once.

The procedure should return a binary string  $S$  of length  $N = 100$  with maximum strength.

## Scoring

If your output does not conform to the constraints described in the implementation details, the score of your solution for that subtask will be 0 (reported as `Output isn't correct` in CMS).

Let  $K$  denote the strength of the string in your output for a given subtask.

In Subtask 3, your score is calculated according to the following table:

Condition	Points
$20 < K$	0
$4 < K \leq 20$	$21 - K$
$K = 4$	20
$K = 3$	25

In Subtask 4, your score is calculated according to the following table:

Condition	Points
$K \leq 50$	0
$50 < K \leq 80$	$K - 50$
$80 < K \leq 83$	$30 + 5 \cdot (K - 80)$
$K = 84$	60

## Sample Grader

Parts I and II use the same sample grader program, with the distinction between the two parts determined by the first line of the input.

Input format for Part I:

```
1
S
```

Output format for Part I:

```
K
```

Input format for Part II:

```
2
T
```

Here,  $T$  is either the string `weakest` or the string `strongest`.

Output format for Part II:

```
S
```

Note that the output of the sample grader adheres to the required format for the output files in Part II.