**Longest repeating and non-overlapping substring**

Given a string **s**of length **n**, find the longest repeating non-overlapping substring in it. In other words find 2 identical substrings of maximum length which **do not overlap**. Return the longest non-overlapping substring. Return **"-1"** if no such string exists.

**Note:**Multiple Answers are possible but you have to return the substring whose **first occurrence is earlier**.

**For Example:** "abhihiab", here both **"ab"** and **"hi"** are possible answers. But you will have to return **"ab"** because it's first occurrence appears before the first occurrence of "hi".

**Example 1:**

**Input:**

n = 9

s ="acdcdacdc"

**Output:**

"acdc"

**Explanation:**

The string "acdc" is the longest Substring of s which is repeating but not overlapping.

**Example 2:**

**Input:**

n = 7

s ="heheheh"

**Output:**

"heh"

**Explanation:**

The string "heh" is the longest Substring of s which is repeating but not overlapping.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function**longestSubstring()** which takes an Integer **n** and a string **s** as input and returns the answer.

**Expected Time Complexity:** O(n2)  
**Expected Auxiliary Space:** O(n2)

**Constraints:**  
1 <= n <= 103

Class Solution {

public:

string longestSubstring(string s, int n) {

// code here

int window\_size=1;

string sub\_str="";

string result;

for(int i=0;i<n;i++){

while(sub\_str.length()!=window\_size){

sub\_str+=s[i];

}

size\_t found=s.find(sub\_str,i+1);

if(found != string::npos){

window\_size++;

result=sub\_str;

}

else{

sub\_str.erase(0,1);

}

}

if(result.length()==0) return "-1";

return result;

}

};

Link : <https://www.geeksforgeeks.org/problems/longest-repeating-and-non-overlapping-substring3421/1>