**FINDING CB NUMBERS**

Deepak and Gautam are having a discussion on a new type of number that they call Coding Blocks Number or CB Number. They use following criteria to define a CB Number.

1. 0 and 1 are not a CB number.
2. 2,3,5,7,11,13,17,19,23,29 are CB numbers.
3. Any number not divisible by the numbers in point 2( Given above) are also CB numbers.

Deepak said he loved CB numbers.Hearing it, Gautam throws a challenge to him.

Gautam will give Deepak a string of digits. Deepak's task is to find the number of CB numbers in the string.

CB number once detected should not be sub-string or super-string of any other CB number.  
Ex- In ****4991****, both ****499**** and ****991**** are CB numbers but you can choose either ****499**** or ****991****, not both.

Further, the CB number formed can only be a sub-string of the string.  
Ex - In ****481****, you can not take ****41**** as CB number because 41 is not a sub-string of ****481****.

As there can be multiple solutions, Gautam asks Deepak to find the maximum number of CB numbers that can be formed from the given string.

Deepak has to take class of Launchpad students. Help him by solving Gautam's challenge.

**Input Format:**

First line contain size of the string.

Next line is A string of digits.

**Constraints:**

1 <= Length of strings of digits <= 17

**Output Format**

Maximum number of CB numbers that can be formed.

**Sample Input**

5

81615

**Sample Output**

2

Program-

#include<iostream>

#include<cstring>

#include "unordered\_map"

using namespace std;

long long int stringtoint(string s,int n)

{

if(n==1)

{

return s[n-1]-'0';

}

return(10\*stringtoint(s,n-1)+s[n-1]-'0');

}

bool checkIfCBNumber(long long int num)

{

if(num==0 || num==1)

return false;

int primes[10]={2,3,5,7,11,13,17,19,23,29};

for(int i=0;i<10;i++)

{

if(num%primes[i]==0 && num != primes[i])

return false;

}

return true;

}

// void removeDups(int arr[], int n)

// {

// unordered\_map<int, bool> mp;

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

// if (mp.find(arr[i]) == mp.end()) {

// checkIfCBNumber(arr[i]);

// }

// mp[arr[i]] = true;

// }

// }

bool is\_valid(bool \*visited, int l, int r){

for(int i = l; i < r; i++)

{

if(visited[i])

return false;

}

return true;

}

int main()

{

int n,i,j,k=0,arr[100];

cin>>n;

bool \*visited = new bool[n];

for(int i = 0; i < n; i++) visited[i] = false;

string s;

cin>>s;

int count = 0;

for(i=1;i<=n;i++)

{

for(j=0;j<=n-i;j++)

{

int end = j+i;

string str = s.substr(j, i);

int size=str.length();

long long int num=stringtoint(str,size);

if(checkIfCBNumber(num) && is\_valid(visited, j, end)){

count++;

for(int k = j; k < end; k++)

visited[k]=true;

}

// arr[k]=num;

// k++;

}

}

// removeDups(arr, k);

cout<<count;

}