Lab6 Solutions

Meow

เขียน function เพื่อตรวจสอบสตริง เปลี่ยนตัวอักษรให้เป็นพิมพ์เล็กให้หมดก่อนเพื่อจะได้ตรวจสอบง่าย จากนั้นลบตัวอักษรซ้ำและอยู่ติดกัน โดยเลือกเฉพาะตัวอักษรที่ตัวก่อนหน้าไม่เท่ากับตัวอักษรตัวนั้นแล้วเก็บในสตริงใหม่(เทียบกับตัวหลังแทนก็ได้ ทั้งสองแบบให้ระวังเรื่องตัวอักษรตัวแรกและตัวสุดท้ายในสตริงเดิม) สตริงใหม่ที่ได้จะเป็นเป็นเสียงร้องของแมวเมื่อเท่ากับ “meow” เท่านั้น

Example Code

```cpp
#include <bits/stdc++.h>
using namespace std;

bool meow(string str,int length)
{
    for(int i=0;i<length;i++) str[i] = tolower(str[i]);

    string temp = "";
    temp += str[0];
    for(int i=1;i<length;i++) if(str[i] != str[i-1]) temp = temp + str[i];

    return temp == "meow";
}

int main()
{
    int n,length;
    cin >> n;
    while(n--)
    {
        string str;
        cin >> length >> str;
        (meow(str,length))? cout << "YES": cout << "NO";
        cout << endl;
    }

}
```

$time\ complexity : O(n \cdot length)$

Prem Sprint

จากโจทย์นายเปรมสามารถเดินไปได้ 4 ทิศทางคือ ขึ้น ลง ซ้าย ขวา ดังนั้นถ้านายเปรมอยู่ที่ตำแหน่ง (i,j) ใดๆ จะเดินต่อไปได้ 4 ตำแหน่งได้แก่ $(i-1,j),(i+1,j),(i,j-1)$ และ $(i,j+1)$ ดังนั้นสามารถสร้าง *recursive function* สำหรับการเดินของนายเปรมได้ โดยรับค่า *(i,j)* แล้วส่งต่อไปยังอีก 4 ตำแหน่งข้างต้น สร้าง *array* $status[n][n]$ เพื่อป้องกัน *infinite recursion* และเก็บข้อมูลว่าตำแหน่งไหนสามารถเดินไปถึงได้บ้าง ให้เริ่มเรียก *function* ครั้งแรกที่ $(0,0)$ เสร็จแล้วตรวจสอบค่าของ $status[n-1][n-1]$ เพื่อตอบว่าสามารถไปถึงห้องน้ำได้หรือไม่

Example Code

```cpp
#include <bits/stdc++.h>
#define MAX 100
using namespace std;

int n, arr[MAX][MAX] = {0};
int status[MAX][MAX] = {0}; // 0->not visited, 1->Yes, 2->No

void rec(int i,int j)
{
    if(i<0 || j<0 || i>=n || j>=n || status[i][j]) return;

    if(arr[i][j]==0) status[i][j] = 1;
    else if(arr[i][j]==1) status[i][j] = 2;

    if(status[i][j]==2) return;
    rec(i-1,j);
    rec(i+1,j);
    rec(i,j-1);
    rec(i,j+1);
}

int main()
{
    cin >> n;
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) cin >> arr[i][j];

    rec(0,0);
    (status[n-1][n-1]==1)? cout << "YES": cout << "NO";
}
```

$time\ complexity : O(n^2)$

Make Prime

พิจารณาตัวเลขใด ๆ จำนวนวิธีในการลบเลขโดดจะเท่ากับจำนวนหลักของเลขนั้น และจำนวนการลบเลขโดดที่น้อยที่สุดให้เป็นจำนวนเฉพาะสามารถคิดได้จาก คำนวณวิธีการลบเลขโดดที่น้อยที่สุดให้เป็นจำนวนเฉพาะของเลขที่ถูกลบไปแล้ว 1 หลักทุกรูปแบบ แล้วนำค่าที่น้อยที่สุดมาบวกเพิ่ม 1 หลักการนี้สามารถนำไปเขียน recursive function ได้ โดยให้หยุดการทำงานเมื่อพบว่าเลขนั้นเป็นจำนวนเฉพาะหรือเลขที่ส่งเข้า function ไปเป็นเลขหลักเดียว ส่วนการตรวจสอบว่าจำนวนเต็ม n เป็นจำนวนเฉพาะหรือไม่ให้ loop จำนวนเต็มตั้งแต่ 2 ถึง $\sqrt{n}$ หากเป็นจำนวนเฉพาะต้องไม่มีจำนวนไหนเลยที่หาร n ลงตัว

Example Code

```cpp
#include <bits/stdc++.h>
using namespace std;

bool isprime(int num)
{
    for(int i=2;i<=sqrt(num);i++) if(num%i==0) return false;
    return (num>=2)? true : false;
}

int makeprime(int num,int step=0)
{
    if(num<10) return (isprime(num))? step:INT_MAX;
    if(isprime(num)) return step;

    int minstep = INT_MAX;
    for(int i=1;i<num;i*=10)
    {
        int temp = num/(10*i)*i+num%i;
        minstep = min(makeprime(temp, step+1),minstep);
    }

    return minstep;
}

int main()
{
    int n;
    cin >> n;
    int ans = makeprime(n);
    (ans == INT_MAX)? cout << -1 : cout << ans;
}
```

$time\ complexity : O(\log(n)!)$