PAT_A1015. Reversible Primes (20)

1.Abstraction：进制转换、质数判断

1.1 Algorithm and idea

```
1.进制转换。
2.质数判断。
```

1.2 Notice

```
1.注意判断输入终止的方法。
    一种是题目给出的终止条件，即输入负数。
    一种是文本文件自然结尾，EOF。
```

2.Problem: Reversible Primes (20)

A *reversible prime* in any number system is a prime whose "reverse" in that number system is also a prime. For example in the decimal system 73 is a reversible prime because its reverse 37 is also a prime.

Now given any two positive integers N (< 105) and D (1 < D <= 10), you are supposed to tell if N is a reversible prime with radix D.

Input Specification:

The input file consists of several test cases. Each case occupies a line which contains two integers N and D. The input is finished by a negative N.

Output Specification:

For each test case, print in one line "Yes" if N is a reversible prime with radix D, or "No" if not.

Sample Input:

```
73 10
23 2
23 10
-2
```

Sample Output:

```
Yes
Yes
No
```

3.Algorithm note

质数的判断

```
bool isPrime(int n){
    if(n<=1) return false;//既不是质数，也不是合数
    else{
        int sq=(int)sqrt(1.0*n);
        for(int i=2;i<=sq;++i){
            if(n%i==0) return false;
        }
        return true;
    }
}
```

十进制转化为相应进制

```
//十进制转化为相应的进制
int convertToRadix(int num,int radix,int *num_array){
    int i=0;
    //注意while循环的条件
    do{
        num_array[i++]=num%radix;
        num/=radix;
    }while(num!=0);
    return i;
}
```

相应进制转化为十进制

```
//这里要注意num_array的存储格式问题，也就是num_array哪一边是高位哪一边是低位。
//这里假设num_array的0位是最高位
//这里的进制转换是一个非常精巧的算法一定要记下来
int convertTo10(int *num_array,int len,int radix){
    int ans=0;
    for(int i=0;i<len;++i){
        ans=ans*radix+num_array[i];
    }
    return ans;
}
```

4. Code

4.1 Edit 0:

4.1.1 Algorithm abstraction

4.1.2 Notice

4.1.3 Code Block

```
#include <cstdio>

bool isPrimer(int n){
    if(n<2) return false;
    else if(n==3||n==2) return true;
    else{
        for(int i=2;i*i<=n;++i){
            if(n%i==0){
                return false;
            }
        }
        return true;
```

```
    }
}

int reverse_num(int n,int radix){
    int sum=0;
    while(n){
        sum=sum*radix+n%radix;
        n/=radix;
    }
    return sum;
}

int main(){
    int value,radix;
    scanf("%d %d",&value,&radix);
    while(value>=0){
        if(isPrimer(value)&&isPrimer(reverse_num(value,radix))){
            printf("Yes\n");
        }
        else{
            printf("No\n");
        }
        scanf("%d %d",&value,&radix);
    }
    return 0;
}
```

4.2 Edit 1:

4.2.1 Algorithm abstraction

4.2.2 Notice

4.2.3 Code Block

```
#include <cstdio>
#include <cmath>

bool isPrime(int n){
    if(n<=1) return false;
    else{
        int sq=sqrt(1.0*n);
        for(int i=2;i<=sq;++i){
            if(n%i==0) return false;
        }
        return true;
    }
}


int num_array[112];
```

```
int R_convertTo10(int *num_array,int len,int radix){
    int ans=0;
    for(int i=0;i<len;++i){
        ans=ans*radix+num_array[i];
    }
    return ans;
}

int convertToRadix(int num,int radix,int *num_array){
    int i=0;
    do{
        num_array[i++]=num%radix;
        num/=radix;
    }while(num!=0);
    return i;
}

int main(){
    int num;
    int radix;

    while(scanf("%d",&num)!=EOF){
        if(num>=0){
            scanf("%d",&radix);
            if(isPrime(num)){
                int len=convertToRadix(num,radix,num_array);
                num=R_convertTo10(num_array,len,radix);
                if(isPrime(num)){
                    printf("Yes\n");
                    continue;
                }
            }
            printf("No\n");
        }
        else{
            break;
        }
    }
    return 0;
}
```

5. Summary