

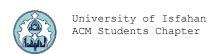
سوال A: <mark>897A</mark>

برای هر کاراکتر i در رشته i در رنج i در رنج i اگر کاراکتر برابر i بود باید با i جایگزین شود. این عملیات باید i مرتبه روی رشته i تکرار شود.

C++:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n, m;
    cin>>n>m;
    string s;
    cin>>s;
    while(m--){
        int l, r;
        char c1, c2;
        cin>>l>>r>>c1>>c2;
        l--, r--;
        for(int i=1; i<=r; i++){
            if(s[i] == c1) s[i]=c2;
        }
    }
    cout<<s;
}</pre>
```

```
n,m=map(int,input().split())
s=input()
for i in range(m):
    L,r,c1,c2=input().split()
    s=s[:int(l)-1]+s[int(l)-1:int(r)].replace(c1,c2)+s[int(r):]
print(s)
```



سوال B: 267A

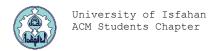
- ❖ همواره b را بزرگتر از a در نظر میگیریم؛ اگر بخواهیم به صورت دستی یکی یکی a را از b کم کنیم تا زمانی که b از a کوچکتر شود، با توجه به محدوده a , a , نامه طولانی خواهد شد.
- ❖ اینجا [b/a] برابر تعداد دفعاتی است که می توان عدد a را از b کم کرد.(تا زمانی که b از a کوچکتر شود.) سپس مقدار جدید b باقی مانده b بر a خواهد بود.(زیرا تا حد امکان b را با پیمانه a کم کرده ایم و آنچه باقی مانده، b ه کا است.) این روند باید ادامه پیدا کند تا وقتی که عدد کوچک تر (a)، صفر شود.

C++:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n;
    cin>>n;
    while(n--){
        int a, b;
        cin>>a>>b;

        int count=0;
        while(b){
            if(a > b){
                swap(a, b);
        }
        count += b/a;
        b = b%a;
    }
    cout<<count<<endl;
}</pre>
```

```
n=int(input())
for i in range(n):
    a,b = map( int, input().split() )
    if a>b:
        a,b=b,a
    ans=0
    while a>0:
        ans+=b//a
        b%=a
        a,b=b,a
    print(ans)
```



- نوض کنید، یک عدد صحیح با بیش از 4 مقسوم علیه داریم (و شرط دیگر بر آورده شده است). اگر a حداقل دو عامل اول متفاوت داشته باشد، می توانیم همه ضرایب اول دیگر را حذف کنیم و عدد کوچک تری با حداقل 4 مقسوم گیرنده به دست آوریم. در غیر این صورت، a=p^k برای مقداری p^3 نیز دارای 4 مقسوم علیه و کوچکتر از a خواهد بود. وقتی یک عامل اول را بیرون می اندازیم، هیچ مقسوم علیه جدیدی ظاهر نمی شود، بنابراین تفاوت بین هر دو از آنها حداقل d خواهد بود.
 - ♦ بیایید کوچکترین اعداد صحیح از فرم 3×p و p^q که p<q را بطور مستقل پیدا کنیم.</p>
- برای p، اولین مقدار که p>=d+1 باشد و برای p، اولین مقدار که q>=p+d باشد را پیدا میکنیم.(چرا این دو مقدار شرط ها را یرای p. برای q.
 بر آورده میکنند؟)
 - ❖ جواب کوچکترین مقدار بین این دو مقدار خواهد بود.

C++:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long 11;
const 11 mx = 1e5+10;
11 is_prime[mx];
std::vector<ll> primes;
int main(){
    for(ll i=2;i<mx;i++){</pre>
        if(is_prime[i]==false){
             primes.push back(i);
             for(ll j=i+i;j<mx;j+=i){</pre>
                 is_prime[j] = true;
    11 t;
    cin >> t;
    while(t--){
        11 \text{ ans} = 2, d, mp = 2;
        cin >> d;
        for(auto p:primes){
             if(p>=d+1){
                 ans = p*p*p;
                 mp = p;
                 break;
         for(auto q:primes){
             if(q>=mp+d){
                 ans = min(q*mp,ans);
                 break;
         cout<<ans<<endl;</pre>
```



```
def foo(n):
    for i in range(2,int(n**0.5//1)+1):
        if(n%i==0):
            return 0
    return 1

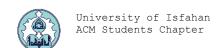
f=int(input())
for q in range(f):
    n=int(input())
    x=n+1
    while(not foo(x)):
        x+=1
    y=x+n
    while(not foo(y)):
    y+=1
    print(x*y)
```

غذاها باید یا دقیقا بعد از بیدار شدن گربه خریده شود، یا سر ساعت 20:00 (اگر که ممکن است.) چون بین ساعت بیدار شدن و ساعت 30:00 قیمت غذا ثابت است ولی میزان گرسنگی گربه افزایش میابد. اگر گربه بعد از ساعت 20:00 بیدار شود، یک انتخاب وجود دارد که همان لحظه غذا بخرد.

C++:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    float a,b,c,d,e,f,g,h;
    cin>>a>>b>>c>>d>e>>f;
    g = ceil(c/f)*e;
    h = ceil((c+d*(60*(19-a)+(60-b)))/f)*e*80/100;
    if(a<20)
    {
        cout<<fixed<<setprecision(4)<<min(g,h)<<endl;
    }
    else
    {
        cout<<fixed<<setprecision(4)<<(g*80/100)<<endl;
    }
    return 0;
}</pre>
```

```
import math
h,m=map(int,input().split())
H,D,C,N=map(int,input().split())
if h>=20:
    print(C*0.8*math.ceil(H/N))
else:
    k=(20-h-1)*60+60-m
    print(min(C*math.ceil(H/N),C*0.8*math.ceil((H+D*k)/N)))
```



حالتی را در نظر بگیرید که حداکثر کاراکتر در رشته در پاسخ باشد. پس پاسخ برابر است با min (r1, r2) - max (11, 12) در غیر این صورت، هر دو رشته را در اطراف این کاراکتر برش می دهیم (در اینجا ممکن است رشته های خالی دریافت کنیم) و الگوریتم را به صورت بازگشتی برای هر جفت رشته ای که به دست می آوریم اجرا می کنیم.

C++:

```
#include<bits/stdc++.h>
using namespace std;
int p(int a,int b,int c,int d,int now){
if(a>b||c>d)return 0;
int l=1<<now,fl=1;</pre>
while(f1){
fl=0;
while(c>l)c-=l,d-=l,fl=1;
while(a>l)a-=l,b-=l,fl=1;
while(l>max(b,d))now--,l>>=1,fl=1;
if(a>c | |a==c&d>b)swap(a,c),swap(b,d);
if(b>=d)return d-c+1;
return max(max(p(a,b,c,l-1,now),p(1,d-1,a,b,now)),b-c+1);
int main(){
int a,b,c,d;
cin>>a>>b>>c>>d;
cout<<p(a,b,c,d,30);
```

```
aaa = 0
def f(l1,r1,l2,r2,top):
    global aaa
    if (l1>r1 or l2> r2):
    if (top-1<=aaa) or (r1-l1+1<=aaa) or (r2-l2+1<=aaa):
        return 0
    if top==2:
        return 1
    if (11>top):
        11 -=top
        r1 -= top
    if (12>top):
        12-=top
        r2-=top
    if (l1==l2 and r1==r2):
        return r1-l1+1
    if (l1==0 and r1==top-1):
```



```
return r2-12+1
    if (12==0 and r2==top-1):
        return r1-l1+1
    if ( (11 \le 12 \text{ and } 12 \le r1) or (12 \le 11 \text{ and } 11 \le r2)):
         ans = min(r1,r2) - max(11,12) +1
    else:
        ans = 0
    top = top//2
    ans = \max(ans, f(11,\min(r1,top-1),12,\min(r2,top-1), top))
    ans = \max(ans, f(11,\min(r1,top-1),\max(top+1,12),r2, top))
    ans = \max(ans, f(\max(11, top+1), r1, 12, \min(r2, top-1), top))
    ans = \max(ans, f(\max(11, top+1), r1, \max(12, top+1), r2, top))
    aaa = max(aaa,ans)
    return ans
a = input().split()
print(f(int(a[0]),int(a[1]),int(a[2]),int(a[3]),2**36))
```