A　计 1——二进制数

Accept:245　　Submit:618

Time Limit:1000MS　　Memory Limit:65536KB

Description

大家都知道，数据在计算机里中存储是以二进制的形式存储的。

有一天，小明学了 C 语言之后，他想知道一个类型为 unsigned int 类型的数字，存储在计算机中的二进制串是什么样子的。

你能帮帮小明吗？并且，小明不想要二进制串中前面的没有意义的 0 串，即要去掉前导 0。

Input

第一行，一个数字 T（T<=1000），表示下面要求的数字的个数。

接下来有 T 行，每行有一个数字 n（0<=n<=10^8），表示要求的二进制串。

Output

输出共 T 行。每行输出求得的二进制串。

Sample Input

5

23

535

2624

56275

989835

Sample Output

10111

1000010111

1010010000000

1101101111010011

11110001101010001011


```
#include<iostream>
using namespace std;
int main(){
    int t;
    int n,i,m;
    int a[40];
    cin>>t;
    while(t--){
        cin>>n;
        i=0;
        while (n!=0){
            a[i]=n%2;
            n=n/2;
            i++;
        }
        m=i-1;
```

```
        for(i=m;i>=0;i--)
            cout<<a[i];
        cout<<endl;
    }
        return 0;
}
```

B　计 2——矩阵幂

Accept:136　　Submit:589

Time Limit:1000MS　　Memory Limit:65536KB

Description

给你一个 n*n 的矩阵，

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix},$$

，

求其矩阵的 k 次幂，即 Pk

Input

第一行，一个整数 T（0<T<=10），表示要求矩阵的个数。

接下来有 T 组数据，每组数据格式如下：

第一行：两个数据 n（2<=n<=10）、k（1<=k<=5），两个数字之间用一个空格隔开，其中 n 表示状况空间的总数，k 表示待求的转移概率矩阵的步数。接下来有 n 行 n 列个正整数，其中，第 i 行第 j 列表示 pij，（0<=pij<=10）。另外，数据保证最后结果不会超过 10^8。

Output

输出为 T 组数据。

每组数据为已知矩阵的 k 次幂，格式为：

n 行 n 列个正整数，每行数之间用空格隔开，注意，每行最后一个数后面不应该有多余的空格。

Sample Input

```
3
2 2
9 8
9 3
3 3
4 8 4
9 3 0
3 5 7
5 2
4 0 3 0 1
```

0 0 5 8 5
8 9 8 5 3
9 6 1 7 8
7 2 5 7 3
Sample Output
153 96
108 81
1216 1248 708
1089 927 504
1161 1151 739
47 29 41 22 16
147 103 73 116 94
162 108 153 168 126
163 67 112 158 122
152 93 93 111 97

```
#include<iostream>
using namespace std;
int main()
{
    int t,n,k,i,j,m;
    int a[11][11];
    int b[11][11];
    int c[11][11];
    cin>>t;
    while(t--){
        cin>>n>>k;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++){
                cin>>a[i][j];
                b[i][j]=a[i][j];
            }
        k--;

        while(k--){
            for(i=0;i<n;i++)
                for(j=0;j<n;j++){
                    c[i][j]=0;
                    for(m=0;m<n;m++){
                        c[i][j]=c[i][j]+a[i][m]*b[m][j];

                        }
                }
```

```
            for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                    b[i][j]=c[i][j];
        }
        for(i=0;i<n;i++){
            for(j=0;j<n-1;j++)
                cout<<b[i][j]<<" ";
            cout<<b[i][n-1]<<endl;
        }
    }
        return 0;
}
```

C　计 3——二叉排序树

Accept:119　　Submit:369

Time Limit:1000MS　　Memory Limit:65536KB

Description

二叉排序树，也称为二叉查找树。可以是一颗空树，也可以是一颗具有如下特性的非空二叉树：

1. 若左子树非空，则左子树上所有节点关键字值均不大于根节点的关键字值；
2. 若右子树非空，则右子树上所有节点关键字值均不小于根节点的关键字值；
3. 左、右子树本身也是一颗二叉排序树。

现在给你 N 个关键字值各不相同的节点，要求你按顺序插入一个初始为空树的二叉排序树中，每次插入后成功后，求相应的父亲节点的关键字值，如果没有父亲节点，则输出-1。

Input

第一行，一个数字 N（N<=100），表示待插入的节点数。

第二行，N 个互不相同的正整数，表示要顺序插入节点的关键字值，这些值不超过 108。

Output

输出共 N 行，每次插入节点后，该节点对应的父亲节点的关键字值。

Sample Input

```
5
2 5 1 3 4
```

Sample Output

```
-1
2
2
5
3
```

```
#include<iostream>
```

```
using namespace std;
typedef struct{
    int l,r,p,m;
}node;
int main(){
    int n,i,j,x,pa;
    node a[120];
    node m;
    while(cin>>n){
        for(i=0;i<n;i++){
            a[i].l=-1;
            a[i].r=-1;
            a[i].p=-1;
            a[i].m=-1;
        }
        for(i=0;i<n;i++){
            cin>>a[i].m;
            if(i==0)
                cout<<-1<<endl;
            else{
                j=0;
                if(a[i].m>a[j].m)
                    x=a[j].r;
                else
                    x=a[j].l;
                while (x!=-1){
                    j=x;
                    if(a[i].m>a[j].m)
                        x=a[j].r;
                    else
                        x=a[j].l;

                }
                a[i].p=j;
                if(a[i].m>a[j].m)
                        a[j].r=i;
                else
                        a[j].l=i;
                cout<<a[a[i].p].m<<endl;
            }
        }
    }
    return 0;
}
```

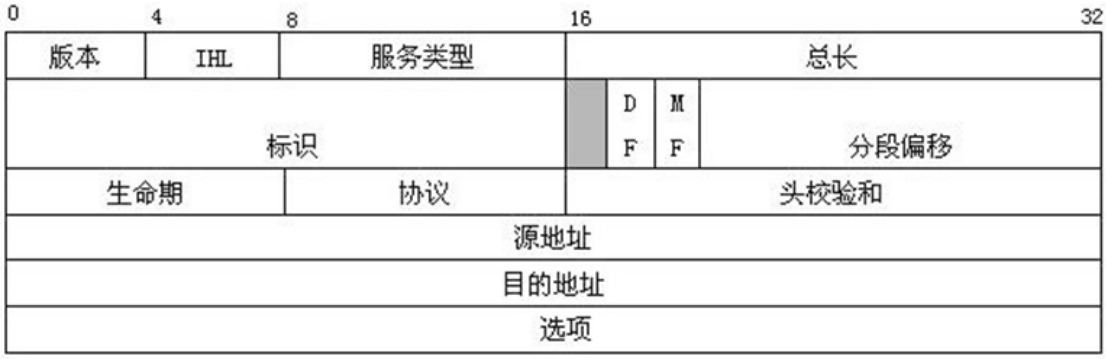
D　计 4——IP 数据包解析

Accept:19　　Submit:194

Time Limit:1000MS　　Memory Limit:65536KB

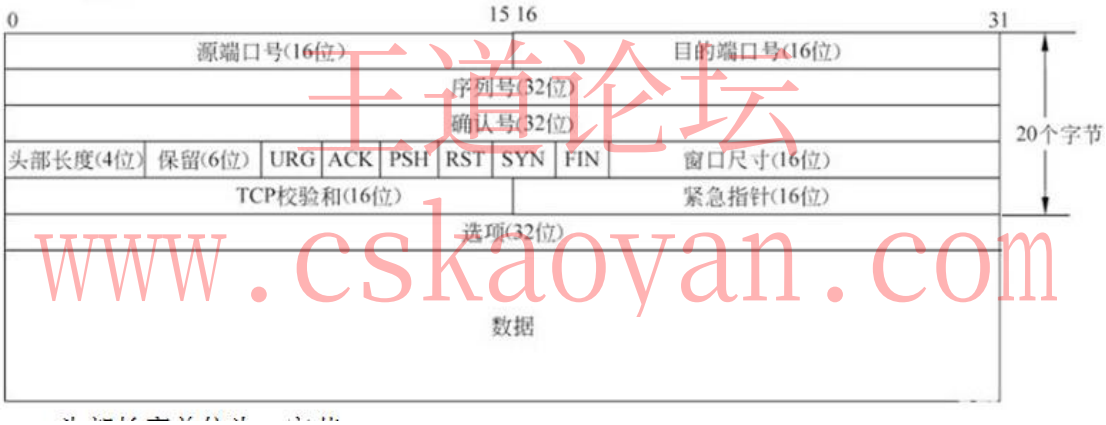
Description

我们都学习过计算机网络，知道网络层 IP 协议数据包的头部格式如下：



其中 IHL 表示 IP 头的长度，单位是 4 字节；总长表示整个数据包的长度，单位是 1 字节。

传输层的 TCP 协议数据段的头部格式如下：



头部长度单位为 4 字节。

你的任务是，简要分析输入数据中的若干个 TCP 数据段的头部。 详细要求请见输入输出部分的说明。

Input

第一行为一个整数 T，代表测试数据的组数。

以下有 T 行，每行都是一个 TCP 数据包的头部分，字节用 16 进制表示，以空格隔开。数据保证字节之间仅有一个空格，且行首行尾没有多余的空白字符。

保证输入数据都是合法的。

Output

对于每个 TCP 数据包，输出如下信息：

Case #x，x 是当前测试数据的序号，从 1 开始。

Total length = L bytes，L 是整个 IP 数据包的长度，单位是 1 字节。

Source = xxx.xxx.xxx.xxx，用点分十进制输出源 IP 地址。输入数据中不存在 IPV6 数据分组。

Destination = xxx.xxx.xxx.xxx，用点分十进制输出源 IP 地址。输入数据中不存在 IPV6 数据分组。

Source Port = sp，sp 是源端口号。
Destination Port = dp，dp 是目标端口号。

对于每个 TCP 数据包，最后输出一个多余的空白行。
具体格式参见样例。
请注意，输出的信息中，所有的空格、大小写、点符号、换行均要与样例格式保持一致，并且不要在任何数字前输出多余的前导 0，也不要输出任何不必要的空白字符。
Sample Input
2
45 00 00 34 7a 67 40 00 40 06 63 5a 0a cd 0a f4 7d 38 ca 09 cd f6 00 50 b4 d7 ae 1c 9b cf f2 40 80 10 ff 3d fd d0 00 00 01 01 08 0a 32 53 7d fb 5e 49 4e c8
45 00 00 c6 56 5a 40 00 34 06 e0 45 cb d0 2e 01 0a cd 0a f4 00 50 ce 61 e1 e9 b9 ee 47 c7 37 34 80 18 00 b5 81 8f 00 00 01 01 08 0a 88 24 fa c6 32 63 cd 8d
Sample Output
Case #1
Total length = 52 bytes
Source = 10.205.10.244
Destination = 125.56.202.9
Source Port = 52726
Destination Port = 80

Case #2
Total length = 198 bytes
Source = 203.208.46.1
Destination = 10.205.10.244
Source Port = 80
Destination Port = 52833


```
#include<iostream>
#include<string>
using namespace std;
int st(char c){
    int a;
    if(c>='0' && c<='9')
            a=(int)(c-'0');
        else
            a=(int)(c-'a'+10);
    return a;
}
int main()
{
    int t,i,k,l,l1;
    char c;
```

```
    string a;
    cin>>t;

    for(k=1;k<=t;){
        getline(cin,a);
    if(a!=""){

        a=a+" ";
        cout<<"Case #"<<k<<endl;
        i=0;
        l=st(a[1]);
         l1=st(a[6])*4096+st(a[7])*256+st(a[9])*16+st(a[10]);
         cout<<"Total length = "<<l1<<" bytes"<<endl;
         cout<<"Source = "<<st(a[36])*16+st(a[37])<<"."<<st(a[39])*16+st(a[40])<<"."<<st(a[42])*16+st(a[43])<<"."<<st(a[45])*16+st(a[46])<<endl;
         cout<<"Destination = "<<st(a[48])*16+st(a[49])<<"."<<st(a[51])*16+st(a[52])<<"."<<st(a[54])*16+st(a[55])<<"."<<st(a[57])*16+st(a[58])<<endl;
         l=12*l;
         cout<<"Source Port = "<<st(a[l])*4096+st(a[l+1])*256+st(a[l+3])*16+st(a[l+4])<<endl;
         cout<<"Destination Port = "<<st(a[l+6])*4096+st(a[l+7])*256+st(a[l+9])*16+st(a[l+10])<<endl<<endl;
         k++;
    }
    }
        return 0;
}
```