结构体

结构体排序参考解法：

```
#include <bits/stdc++.h>//1-1414-1 推荐学习    javacn
using namespace std;
struct stu{
    int num;
    string name;
    int score;
}a[110];
// 按成绩降序，成绩相同按学号升序
bool cmp(stu s1,stu s2){
      if(s1.score > s2.score || (s1.score == s2.score && s1.num < s2.num))
      {
            return true;
      }
      else{ return false;     }
}
int main()
{
      int n,i;
      cin>>n;
      for(i = 0;i < n;i++)
      {
            cin>>a[i].num>>a[i].name>>a[i].score;
      }
      sort(a,a+n,cmp);

      // 输出
      for(i = 0;i < n;i++)
      {
            cout<<a[i].num<<" "<<a[i].name<<" "<<a[i].score<<endl;
      }
}
```

期末考试成绩排名

冒泡排序参考解法：

```
#include<bits/stdc++.h>//1-1414-2    javacn
using namespace std;
int a[101],c[101];
string b[101];
int main(){
	int n,i,j;
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>a[i]>>b[i]>>c[i];
	}
	// 冒泡排序
	for(i=1;i<=n-1;i++)
	{
		for(j=1;j<=n-i;j++)
		{
			if(c[j] < c[j+1] || c[j]==c[j+1] && a[j]>a[j+1])
			{
				swap(a[j],a[j+1]);
				swap(b[j],b[j+1]);
				swap(c[j],c[j+1]);
			}
		}
	}

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

```c
#include<stdio.h>//1-1414-3  475214719
struct student{
	int xh;
	char xm[20];
	int cj;
};
int main()
{
	int i,j,n;
	struct student t;
	scanf("%d",&n);
	struct student stu[n];
	for(i=0;i<n;i++)
	    scanf("%d%s%d",&stu[i].xh,&stu[i].xm,&stu[i].cj);
	for(i=0;i<n;i++)
		for(j=0;j<n-i-1;j++)
			if(stu[j].cj<stu[j+1].cj)
			{
				t=stu[j];
				stu[j]=stu[j+1];
				stu[j+1]=t;
			}

	for(i=0;i<n;i++)
		for(j=0;j<n-i-1;j++)
			if(stu[j].cj==stu[j+1].cj&&stu[j].xh>stu[j+1].xh)
			{
				t=stu[j];
				stu[j]=stu[j+1];
				stu[j+1]=t;
			}

	for(i=0;i<n;i++)
	    printf("%d %s %d\n",stu[i].xh,stu[i].xm,stu[i].cj);
}
```

```c
#include<stdio.h>//1-1414-4   475214719
struct student
{
	int xh;
	char xm[20];
	int cj;
};
int main()
{
	int i,j,n;struct student t;
	scanf("%d",&n);
	struct student stu[n];

	for(i=0;i<n;i++)
	    scanf("%d%s%d",&stu[i].xh,&stu[i].xm,&stu[i].cj);

	for(i=0;i<n;i++)
	    for(j=0;j<n-i-1;j++)
	        if(stu[j].cj<stu[j+1].cj||stu[j].cj==stu[j+1].cj&&stu[j].
xh>stu[j+1].xh)
	        {
	            t=stu[j];
	            stu[j]=stu[j+1];
	            stu[j+1]=t;
	        }
	for(i=0;i<n;i++)
	    printf("%d %s %d\n",stu[i].xh,stu[i].xm,stu[i].cj);
}
```

坐标排序

```
#include<bits/stdc++.h>//2-1490    wupeng
using namespace std;
struct node{
	int x;
	int y;
}a[10010];
int n;

bool cmp(node a, node b){
	if(a.x==b.x)
	{
		return a.y<b.y;
	}
	else
	{
		return a.x<b.x;
	}
}

int main(){
	cin >> n;
	for(int i=1; i<=n; i++)
	{
		cin >> a[i].x >> a[i].y;
	}

	sort(a+1, a+1+n, cmp);

	for(int i=1; i<=n; i++)
	{
		cout << a[i].x << ' '<< a[i].y << endl;
	}
	return 0;
}
```

解法二：结构体数组求解

```
#include<bits/stdc++.h>//3-1315-1  推荐学习    javacn
using namespace std;
struct node{// 思路：用结构体数组存储每个同学的学号、平均分
      int num;// 编号
      double score;// 平均分
};
node a[110];
int n;
int s,ma,mi;// 存储每个人的总分、最高、最低分
int x;// 表示每个人的 5 个分数
bool cmp(node n1,node n2){// 排序
      return n1.score > n2.score;// if(n1.score>n2.score) return true;
} //  else return false;
int main(){
      cin>>n;
      for(int i = 1;i <= n;i++)
      {     cin>>a[i].num;// 编号
            s = 0;// 总分
            ma = INT_MIN;// 最高
            mi = INT_MAX;// 最低
            for(int j = 1;j <= 5;j++)            // 读入 5 个成绩
            {     cin>>x;
                  s = s + x;// 求和
                  ma = max(ma,x);
                  mi = min(mi,x);
            }
            a[i].score = (s - ma - mi) / 3.0;          // 平均分
      }
      sort(a+1,a+n+1,cmp);     // 排序
      for(int i = 1;i <= 3;i++){
            cout<<a[i].num<<" ";     // 输出
            cout<<fixed<<setprecision(3)<<a[i].score<<endl;       }
    return 0;
}
```

遥控飞机争夺赛

```
#include<bits/stdc++.h>//3-1315-2  仅作参考 liuzhong
using namespace std;
struct stu{
	int a;
	float cj[5];
	float pcj;
}s[100];
bool cmp(stu a,stu b) { return a.pcj>b.pcj;  }
int main()
{
	int n;
	cin>>n;
	float maxx,minn;
	for(int i=0;i<n;i++)
	{
		cin>>s[i].a;
		for(int j=0;j<5;j++)  {	cin>>s[i].cj[j];  }
		maxx=0;
		minn=s[i].cj[0];
		for(int o=0;o<5;o++){
			if(s[i].cj[o]>maxx)	{	maxx=s[i].cj[o];  }
			if(s[i].cj[o]<minn)	{	minn=s[i].cj[o];  }
		}
		s[i].pcj=(s[i].cj[0]+s[i].cj[1]+s[i].cj[2]+s[i].cj[3]+s[i].cj[4]-
minn-maxx)/3.0;
	}
	sort(s,s+n,cmp);
	cout<<s[0].a<<" ";
	printf("%.3f\n",s[0].pcj);
	cout<<s[1].a<<" ";
	printf("%.3f\n",s[1].pcj);
	cout<<s[2].a<<" ";
	printf("%.3f\n",s[2].pcj);
}
```

解法一：二维数组求解

```
#include<bits/stdc++.h>//3-1315-3 仅作参考   javacn
using namespace std;
int main(){
	int n,i,j,t,max,min,a[110][7];
	cin>>n;
	for(i=0;i<n;i++){
		// 当前行的总分
		t = 0,max = 1,min = 1;// 此处写的不妥当
		cin>>a[i][0];// 编号
		for(j=1;j<=5;j++){
			cin>>a[i][j];
			t = t + a[i][j];
			if(a[i][j]>a[i][max]){  max = j;    }
			if(a[i][j]<a[i][min]){  min = j;    }
		}
		a[i][6] = t - a[i][max] - a[i][min];
	}
	for(i=1;i<=n;i++){      // 比较成绩   联动编号
		for(j=0;j<n-i;j++){
			if(a[j][6] < a[j+1][6]){
				t = a[j][6];
				a[j][6] = a[j+1][6];
				a[j+1][6] = t;
				t = a[j][0];
				a[j][0] = a[j+1][0];
				a[j+1][0] = t;
			}
		}
	}
	for(i=0;i<3;i++){
		cout<<a[i][0]<<" "<<fixed<<setprecision(3)<<a[i][6]/3.0<<endl;
	}
	return 0;
}
```

分别建两个数组，保存编号和平均值

```
#include <bits/stdc++.h>//3-1315-4    jiangyf70
using namespace std;
int a;
int b[110]; // 存编号
double c[110]; // 存平均值
int main(){
    int n;
    cin >> n;
    for(int i = 0; i < n; i++)
    {
        int maxn = 0, minn = 999, sum = 0;
        cin >> b[i]; // 读入编号
        for(int j = 0; j < 5; j++)
        {
            cin >> a;
            sum += a;
            if(maxn < a) maxn = a;
            if(minn > a) minn = a;
        }
        c[i] = (sum - maxn - minn) / 3.0; // 求平均值
    }
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j <= n - i; j++)
        {
            if(c[j] < c[j+1]){
                swap(c[j], c[j+1]);  // 冒泡排序
                swap(b[j], b[j+1]);
            }
        }
    }
    for(int i = 0; i < 3;i++){ printf("%d %.3lf\n", b[i], c[i]);}// 输出前三位
      return 0;
}
```

解法二：使用结构体求解

```
#include<bits/stdc++.h>//4-1730-1 推荐学习 javacn
using namespace std;
struct node{
	int money,num;
};
node a[1010];
int m,n;
// 按单价升序排序
bool cmp(node n1,node n2){
	return n1.money<n2.money;
}
int main()
{
	cin>>m>>n;
	for(int i=1;i<=n;i++){
		cin>>a[i].money>>a[i].num;
	}
	sort(a+1,a+n+1,cmp);
	int c=0,s=0;
	for(int i=1;i<=n;i++)
	{
		if(c+a[i].num<m)
		{	// 这家店都买完不够
			c=c+a[i].num;
			s=s+a[i].num*a[i].money;
		 }
		else
		{
			s=s+(m-c)*a[i].money; // 买够 m
			break;
		 }
	 }
	 cout<<s;
}
```

本题显然是使用贪心的思想，既然要花最少的钱购买贺卡，那么必然优先购买价格低的贺卡。将所有店铺的贺卡按价格降序，但要注意数据上要保证价格和数量还是要对应的。

然后从第一家店，依次购买，直到数量满足 m。

购买贺年卡

解法一：使用数组直接排序。

```
#include <bits/stdc++.h>//4-1730-2    javacn
using namespace std;
int m,n;
int price[1100],num[1100],i,j,s,c;
int main() {
	cin>>m>>n;  //m:要买的数量 n:有多少商家
	for(i = 0;i < n;i++){   cin>>price[i]>>num[i];} //读入每家的单价和存货量
	//对单价升序排序，交换单价的同时交换数量
	//排序的次数
	for(i = 1;i <= n - 1;i++){
		for(j = 0;j <= n - i - 1;j++){
			if(price[j] > price[j+1]){
				swap(price[j],price[j+1]);
				swap(num[j],num[j+1]);
			}
		}
	}
	//从第1家开始买
	c = 0; //买了多少
	s = 0;//花了多少钱
	for(i = 0;i < n;i++){
		//先买后退
		c = c + num[i];
		s = s + num[i] * price[i];
		if(c >= m){       //判断要不要退（多买或者正好，循环停止）
			//退掉多买的
			s = s - (c - m) * price[i];
			break;
		}
	}
	cout<<s;
	return 0;
}
```

```c
#include <stdio.h>//4-1730-3    仅作参考  w2ql
void sort(int dj[],int kc[],int n);
int main(){
	int m,n;//m 张贺卡，n 个店铺
	int dj[101],kc[1001];
	scanf("%d%d",&m,&n);
	int i,j;
	for(i=1;i<=n;i++){
		scanf("%d%d",&dj[i],&kc[i]);
	}
	// 按照单价从低到高排序
	sort(dj,kc,n);
	// 尽量从单价最低的购买
	int je=0;// 金额
	for(i=1;i<=n;i++){
		// 购买策略 :kc[i]>=m, 则购买 m 张，结束购买
		//kc[i]<m 则购买 kc[i] 张
		if(kc[i]<m){
			m-=kc[i];// 购买 kc[i] 张
			je+=dj[i]*kc[i];
			kc[i]=0;
		}else{
			kc[i]-=m;// 减库存
			je+=dj[i]*m;// 累加金额
			m=0;// 购买 m 张
			break;
		}
	}
	printf("%d\n",je);
	/*for(i=1;i<=n;i++){
		printf( “%d  “,kc[i]);
	}*/

	return 0;
}
```

```
void sort(int dj[],int kc[],int n){
            int i,j;
            for(i=1;i<=n-1;i++){
                  for(j=1;j<=n-i;j++){
                        if(dj[j]>dj[j+1]){
                              int t=dj[j];
                              dj[j]=dj[j+1];
                              dj[j+1]=t;

                              t=kc[j];
                              kc[j]=kc[j+1];
                              kc[j+1]=t;
                        }
                  }
            }
}
```

```cpp
#include<bits/stdc++.h>//5-1499-1 结构体数组求解法： 推荐学习 javacn
using namespace std;
struct node {
	string name;
	int num;
} a[1010];
bool cmp(node a,node b) {// 按得票降序，得票相同按名字降序
	if(a.num > b.num || a.num == b.num && a.name > b.name) {return true;}
	else {return false;}
}
int main() {
	int n,k=0;//k 代表结构体数组的下标
	bool f=false;
	cin>>n;
	string s;// 每次得票的人
	for(int i=1; i<=n; i++) {
		f=false;
		cin>>s;
		// 在数组中找一下有没有这个人
		for(int j=1; j<=k; j++) {
			// 找到，增加得票数
			if(a[j].name==s) {
				f=true;
				a[j].num++;
			}
		}
		if(f==false) {                  // 没找到，将该人存入数组
			k++;
			a[k].name=s;
			a[k].num=1;        }
	}
	sort(a+1,a+k+1,cmp);     // 排序
	for(int i=1; i<=k; i++) {      cout<<a[i].name<<" "<<a[i].num<<endl;      }
	return 0;
}
```

宇宙总统 2

```
#include<bits/stdc++.h>//5-1499-2  dragoncatter
using namespace std;
struct node
{
	string s;
	int num;
}a[1010];

int  n;
map<string , int> f;
bool cmp(node a, node b)
{
	if(a.num != b.num) return a.num > b.num;
	else return a.s > b.s;
}
int main()
{
	cin>>n;
	for(int i=1;i<=n;i++)
	{
		cin>>a[i].s;
		f[a[i].s]++;
		a[i].num = f[a[i].s];
	}
	sort(a+1,a+n+1,cmp);
	for(int i= 1;i<=n;i++)
	{
		if(f[a[i].s])
		{
			cout<<a[i].s<<' '<<a[i].num<<endl;
			f[a[i].s] = 0;
		}
	}
}
```

解法二：使用结构体解题

```
#include <bits/stdc++.h>//6-1372-1    推荐学习    javacn
using namespace std;
// 每个时间对
struct node {
	int begin;
	int end;
};
struct node a[110];
int n;
// 按结束时间升序排序
bool cmp(node n1,node n2) {
	return n1.end < n2.end?true:false;
}
int main() {
	int i;
	cin>>n;
	for(i = 0; i < n; i++)
	{
		cin>>a[i].begin>>a[i].end;
	}
	sort(a,a+n,cmp);
	int ans=1;  // 第 1 个活动必选
	int t = a[0].end;
	for(int i=1; i<n; i++)  // 在剩余活动中选择
	{
		// 如果当前活动与之前最后结束的活动不冲突，就接受当前活动。
		if(a[i].begin>=t)
		{
			ans++;
			t=a[i].end;
		}
	}
	cout<<ans;
}
```

贪心的策略：优先选择结束时间尽可能早的活动，因为结束时间越早，剩余的时间就越多，那么剩余的时间可以排更多的活动．解法：按照结束时间升序排序，然后逐个统计哪个活动的开始时间 >= 上一个被选中活动的结束时间　　解法一：使用整数数组排序解题

```
#include <bits/stdc++.h>//6-1372-2  javacn
using namespace std;
int n;// 活动数量
int s[110],e[110];// 活动起止时间
int i,j,c,etime;
int main() {
	cin>>n;
	for(i = 0;i < n;i++)	{	cin>>s[i]>>e[i];	}
	for(i = 1;i <= n - 1;i++)// 按照结束时间升序
	{
		for(j = 0;j <= n - i - 1;j++)
		{
			if(e[j] > e[j+1])
			{
				swap(e[j],e[j+1]);
				swap(s[j],s[j+1]);
			}
		}
	}
	etime = e[0];// 存储第 1 个活动的结束时间	// 第 1 个活动必选
	c = 1;// 已经有 1 个选中的活动了
	for(i = 1;i < n;i++)	// 在剩余活动中继续选
	{
		if(s[i] >= etime)// 如果活动的开始时间 >= 上一个选中活动的结束时间
		{
			c++;
			etime = e[i];
		}
	}
	cout<<c;
	return 0;
}
```

活动选择

```
#include<bits/stdc++.h>//6-1372-3    仅作参考 jiangyf70
using namespace std;
struct meet
{
    int l, r;
}a[105];

bool cmp(meet a, meet b)
{
    if(a.r < b.r) return 1;
    return 0;
}
int main()
{
    int n;
    cin >> n;
    for(int i = 0; i < n; i++) cin >> a[i].l >> a[i].r;
    sort(a, a+n, cmp);
    int cnt = 0;
    for(int i = 1; i < n; i++)
    {
        int j = i;
        while(a[i].r > a[j].l) j++;
        cnt++;
        i = j - 1;

    }
    cout << cnt;
    return 0;
}
```

```cpp
#include<bits/stdc++.h>//7-1740-1  推荐学习 jiangyf70
using namespace std;
struct num
{
    int k, c;
}a[105];
bool cmp(num a, num b)
{
    if(a.k < b.k)return 1;
    return 0;
}
int main()
{
    int n;
    cin >> n;
    int m = 0, x;
    for(int i = 0; i < n; i++)
    {
        cin >> x;
        bool f = 0;
        for(int j = 0; j < m; j++)
        {
            if(a[j].k == x)    {    a[j].c++;    f = 1;    }
        }
        if(f == 0)    {    a[m].k = x;    a[m].c = 1;    m++;    }
    }

    sort(a, a + m, cmp);
    for(int i = 0; i < m; i++)
    {
        cout << a[i].k << " "<< a[i].c << endl;
    }
    return 0;
}
```

```cpp
#include<bits/stdc++.h>//7-1740-2    w2016010182
using namespace std;
int main()
{
	map<int,int,less<int> >prap;
	int n;
	scanf("%d",&n);
	for(int i=1;i<=n;++i)
	{
		int dup;
		scanf("%d",&dup);
		if(prap.count(dup)==0)
		{
			pair<int,int>prap2(dup,1);
			prap.insert(prap2);
		}
		else
		{
			prap[dup]+=1;
		}
	}
	map<int,int>::iterator it;
	for(it=prap.begin();it!=prap.end();it++)
	{
		printf("%d %d\n",it->first,it->second);
	}
	return 0;
}
```

统计每个数出现的次数

思路：将数组排序，统计连续相同的数有几个

1. 遇到每个数都 c++

2. c++ 之后判断连续相同的数是否结束

结束标准：到了最后一个数 || 当前数 != 下一个数

```
#include <bits/stdc++.h>//7-1740-3  javacn
using namespace std;
/*
思路：将数组排序，统计连续相同的数有几个
1. 遇到每个数都 c++
2. c++ 之后判断连续相同的数是否结束
结束标准：到了最后一个数 || 当前数 != 下一个数
*/
int n,a[1010],c = 0;
int main()
{
	cin>>n;
	for(int i = 0;i < n;i++){
		cin>>a[i];
	}
	// 排序
	sort(a,a+n);
	// 统计连续相同的数出现的次数
	for(int i = 0;i < n;i++)
	{
		c++;// 统计连续相同的数出现的次数
		if(i == n - 1 || a[i] != a[i + 1])  // 判断连续相同的数是否结束
		{
			cout<<a[i]<<" "<<c<<endl;	// 输出
			c = 0;		// 计数器清零
		}
	}
	return 0;
}
```

混合牛奶 Mixing Milk 按单价排序结构体，从单价最小的开始采购，直到数量 sum==n 即可

```
#include<bits/stdc++.h>//8-1940    jiangyf70
using namespace std;
struct niu
{
    int p, a;
}a[2000005];
bool cmp(niu a, niu b)
{    return a.p < b.p;}
int main()
{
    int n, m;
    cin >> n >> m;
    for(int i = 0; i < m; i++)  {    cin >> a[i].p >> a[i].a;   }
    sort(a, a + m, cmp);
    int sum = 0, fy = 0, i = 0;
    while(sum < n)
    {
        if(sum + a[i].a <= n)
        {
            fy += a[i].a * a[i].p;
            sum += a[i].a;
        }
        else
        {
            fy = fy + (n - sum) * a[i].p;
            sum = n;
            break;
        }
        i++;
    }
    cout << fy;
    return 0;
}
```

仰卧起坐成绩统计

结构体求解：

```
#include<bits/stdc++.h>//9-1314-1 推荐学习 javacn
using namespace std;
struct node {
	int cnt;
	char grade;
};
node a[10];
int n,x;
bool cmp(node n1,node n2) {
	if(n1.cnt>n2.cnt||(n1.cnt==n2.cnt&&n1.grade<n2.grade)) {	return true;}
	else {return false;	}
}
int main() {
	cin>>n;
	string t="0ABCDEFG";
	for(int i=1; i<=6; i++) {
		a[i].grade=t[i];
		a[i].cnt=0;
	}
	for(int i=1; i<=n; i++)
	{
		cin>>x;
		if(x>=60)a[1].cnt++;
		else if(x>=50)a[2].cnt++;
		else if(x>=40)a[3].cnt++;
		else if(x>=30)a[4].cnt++;
		else if(x>=20)a[5].cnt++;
		else a[6].cnt++;
	}
	sort(a+1,a+6+1,cmp);
	for(int i=1; i<=6; i++) {	cout<<a[i].grade<<":"<<a[i].cnt<<endl;}
	return 0;
}
```

结构体无函数求解

```
#include<bits/stdc++.h>//9-1314-2  推荐学习  475214719
using namespace std;
struct t {
    char ch;
    int cnt;
} s[6] = {'A', 0, 'B', 0, 'C', 0, 'D', 0, 'E', 0, 'F', 0};

int main() {
      int n, a[101];
      cin >> n;
      for(int i = 1; i <= n; i++)
            cin >> a[i];

      for(int i = 1; i <= n; i++)
      {
            if(a[i] >= 60)       s[0].cnt++;
            else if(a[i] >= 50) s[1].cnt++;
            else if(a[i] >= 40) s[2].cnt++;
            else if(a[i] >= 30) s[3].cnt++;
            else if(a[i] >= 20) s[4].cnt++;
            else                s[5].cnt++;
      }

      for(int k = 1; k < 6; k++)
            for(int j = 0; j < 6 - k; j++)
                  if(s[j].cnt < s[j + 1].cnt || s[j].cnt == s[j + 1].cnt &&
s[j].ch > s[j + 1].ch)
                        swap(s[j], s[j + 1]);

      for(int i = 0; i < 6; i++)
            cout << s[i].ch <<":" << s[i].cnt << endl;
    return 0;
}
```

新生舞会

```cpp
#include<bits/stdc++.h>//10-1953-1  推荐学习 jiangyf70
using namespace std;
int n; // 人员数量
struct person
{
    string name;
    string num;
    char c;
}a[1005];
char findp(string s)
{
    for(int i = 0; i < n; i++)
    {
        if(a[i].name == s || a[i].num == s) return a[i].c;
    }
    return '0';
}
int main()
{
    cin >> n;
    for(int i = 0; i < n; i++)  {    cin >> a[i].name >> a[i].num >> a[i].c;   }
    int m;
    cin >> m;
    for(int i = 0; i < m; i++)
    {
        string x, y;
        cin >> x >> y;
        if(findp(x) != findp(y)) cout << "Y" << endl;
        else cout << "N" << endl;
    }
    return 0;
}
```

解题思路：用结构体数组读入 n 个人的信息

m 次询问，每询问一次，就判断一次

定义函数，给定一个字符串，函数中判断字符串是名字还是学号

根据给定的字符串，查找其性别

```
#include <bits/stdc++.h>//10-1953-2    javacn
using namespace std;
// 定义结构体的同时，定义结构体数组
struct stu{
	string name;
	string num;
	char gender;
}a[1010];
int n,m;
string s1,s2;
// 根据给定的信息，查询性别
char fun(string s){
	// 如果是学号
	if(isdigit(s[0]))
	{
		for(int i = 1;i <= n;i++)
		{
			if(a[i].num == s) return a[i].gender;
		}
	}
	else
	{
		// 姓名
		for(int i = 1;i <= n;i++)
		{
			if(a[i].name == s) return a[i].gender;
		}
	}
}
```

```
int main()
{
	cin>>n;
	for(int i = 1;i <= n;i++)
	{
		cin>>a[i].name>>a[i].num>>a[i].gender;
	}

	// 读入 m 对要判断的信息
	cin>>m;
	for(int i = 1;i <= m;i++)
	{
		cin>>s1>>s2;
		if(fun(s1) != fun(s2))
		{
			cout<<"Y"<<endl;
		}
		else
		{
			cout<<"N"<<endl;
		}
	}
	return 0;
}
```

进制转换

10 进制和 r 进制互转

正整数 N 转换成一个二进制数

```
#include<iostream>//1-1108-1  jiangyf70
using namespace std;
string i2b(int n, string s)
{
    if(n)
    {
        s = char(n % 2 + '0') + s;
        return i2b(n / 2, s);
    }
    return s;

}

int main()
{
    int n;
    cin >> n;
    if(n == 0) cout << 0;
    else cout << i2b(n,"");
    return 0;
}
```

除 2 取余，结果倒过来连成字符串：

```
#include<bits/stdc++.h>//1-1108-2  javacn
using namespace std;

//10 进制转 2 进制
int n;
string r = "";// 存放转换结果
int main(){
      cin>>n;
      // 当 n!=0 循环
      while(n != 0){
            //cout<<n%2;// 取余
            // 将 n%2 的结果转换为字符 + 到 r 前面
            r = char(n%2+'0') + r;
            n=n/2;// 除 2
      }

      // 特判输入为 0 的情况
      if(r == "") cout<<0;
      else cout<<r;
    return 0;
}
```

二进制转换十进制

按权展开：

```
#include<bits/stdc++.h>//2-1290  javacn
using namespace std;
int main() {
	string s;
	cin>>s;
	int len = s.size();
	long long sum = 0,t = 1;//t 代表权重
	//11010[2 进制 ]=1X2^4+1X2^3+0X2^2+1X2^1+0X2^0=26[10 进制 ]
	// 二进制数转换成十进制数的方法是按权展开
	//2^4 表示 2 的 4 次方
	for(int i=len-1; i>=0; i--) {
		sum = sum + (s[i]-'0') * t;
		t = t * 2;
	}
	cout<<sum;
	return 0;
}
```

正整数 n 转换为 16 进制

除 16 取余，结果倒过来连成字符串：

```
#include<bits/stdc++.h>//3-1289  javacn
using namespace std;

long long n;
string r = "";// 存放 16 进制
string t = "0123456789ABCDEF";// 存放 0~15 对应的字符

int main(){
	cin>>n;
	// 特判输入为 0 的情况
	if(n == 0){
		cout<<0;
		return 0;// 停止函数
	}

	// 短除法
	while(n != 0){
		//n%16：如果在 0~9 之间，转换为字符 '0'~'9'
		// 如果在 10~15 之间，转换为字符 'A'~'F'
		r = t[n%16] + r;
		n = n / 16;// 除 16
	}

	cout<<r;
    return 0;
}
```

十六进制转十进制

```cpp
#include<iostream>//4-1292-1  jiangyf70
using namespace std;
int main()
{
    string s;
    cin >> s;
    long long n = 0, k = 1;;
    for(int i = s.size() - 1; i >= 0; i--)
    {
        if(s[i] - '0' <= 9) n = n + (s[i] - '0') * k;
        else n = n + (s[i] - 55) * k;
        k *= 16;
    }
    cout << n;
    return 0;
}
```

思路：逆序计算，按权展开！

从 s 中获取每一位 s[i] 是字符，要转换为实际的整数！

s[i]: '0'~'9', s[i] - '0'

s[i]: 'A'~'F', s[i] - 'A' + 10

```
#include<bits/stdc++.h>//4-1292-2  javacn
using namespace std;
string s;
long long r = 0,t = 1;//t 表示权重（16 的次方）
int main() {
	cin>>s;
	// 逆序循环字符串，从最低位开始计算
	for(int i = s.size() - 1;i >= 0;i--){
		// 如果当前位是数字字符 0~9
		if(isdigit(s[i])){
			r = r + (s[i] - '0') * t;
		}else{
			// 如果是字母 10~15
			r = r + (s[i] - 55) * t;
		}

		// 权重提升
		t = t * 16;
	}

	cout<<r;
	return 0;
}
```

正整数 n 转换为 8 进制

短除法：除 8 取余，注意本题的 n 要用 long long。

```
#include <bits/stdc++.h>//5-1288 javacn
using namespace std;

long long n;
string r = "";// 存储 n 对应的 8 进制

int main(){
	cin>>n;
	while(n != 0){
		// 倒过来连成字符串
		r = char(n%8+'0') + r;
		n = n / 8;
	}

	if(r == "") cout<<0;
	else cout<<r;
	return 0;
}
```

八进制转十进制

按权展开：

```
#include<bits/stdc++.h>//6-1291   javacn
using namespace std;
int main() {
	// 只能以字符串类型读入
	string s;
	cin>>s;
	int len = s.size();
	long long sum = 0,t = 1;//t 代表权重
	// 类似于 2 进制转 10 进制    8 进制转 10 进制权重为 8
	for(int i=len-1; i>=0; i--) {
		sum = sum + (s[i]-'0') * t;
		t = t * 8;
	}
	cout<<sum;
	return 0;
}
```

小丽找潜在的素数

分别定义函数：判断素数、将二进制转十进制。

```
#include <bits/stdc++.h>//7-1405    javacn
using namespace std;
// 定义函数判断素数
bool sushu(int n){
	bool r = true;// 假设是素数
	for(int i = 2; i <= sqrt(n);i++){
		if(n % i == 0){
			r = false;
			break;
		}
	}
	if(n <= 1) r = false;    // 特判特殊情况
	return r;
}
int jinzhi(string s){// 将二进制转十进制
	int r = 0,t = 1;
	for(int i = s.size()-1;i>=0;i--){    // 倒过来计算，按权展开
		r = r + (s[i] - '0') * t;
		t = t * 2;
	}
	return r;
}
int n,c = 0;
string s;
int main(){
	cin>>n;
	for(int i = 1;i <= n;i++){
		cin>>s;
		if(sushu(jinzhi(s))==true){    c++;   }
	}
	cout<<c;
	return 0;
}
```

小 X 转进制

定义函数判断整数 x 在 m 进制下是否是回文。

```
#include <bits/stdc++.h>//8-1547  javacn
using namespace std;
int n,m,c = 0;
bool huiwen(int x){
	// 将 x 转换为 m 进制
	string t = "0123456789ABCDEF";
	string r = "";
	while(x != 0){
		r = t[x%m] + r;
		x = x / m;
	}
	// 判断 r 是回文
	string r2 = r;
	reverse(r.begin(),r.end());

	if(r2 == r){	return true;	}
	else{ return false;	}
}

int main(){
	cin>>n>>m;
	// 循环 1~n，判断每个数的平方在 m 进制下是否回文
	for(int i = 1;i <= n;i++){
		if(huiwen(i*i)==true){
			c++;
		}
	}

	cout<<c;
	return 0;
}
```

```
/*
1 到 N 中有多少个整数的平方在 M 进制下是回文数呢

思路：
1. 将每个数的平方，转 M 进制
2. 判断回文
*/
```

10 进制转 D 进制

```
#include <bits/stdc++.h>//9-1415-1  yhh133
#define ll long long
using namespace std;
string change(ll n,int x){
	string s;
	while(n){
		if(n%x>=10) s+=n%x-10+'A';
		else s+=n%x+'0';
		n/=x;
	}
	reverse(s.begin(),s.end());
	return s;
}
int main() {
	ll n;
	int k;
	cin>>n>>k;
	if(n==0) cout<<0;
	else cout<<change(n,k);
	return 0;
}
```

除 D 取余法：

```
#include <bits/stdc++.h>//9-1415-2  javacn
using namespace std;
string s = "0123456789ABCDEF",r;
// 短除法求解的余数从 s 中取  例如余数 3，则为 s[3]
int n,d;
int main() {
	cin>>n>>d;
	if(n == 0) {
		cout<<0;
		return 0;
	}
	while(n != 0) {
		r = s[n%d] + r;
		n = n / d;
	}
	cout<<r;
	return 0;
}
```

2 进制和 8、16 进制互转

二进制转换八进制

```
#include<bits/stdc++.h>//1-1293  javacn
using namespace std;
int main(){
	string s;
	cin>>s;
	// 使用循环在字符串 s 前面补 0 使得字符串 s 的长度是 3 的倍数
	while(s.size()%3!=0){
		s = '0' + s;
	}
	int len = s.size(),i;
	//3 个 2 进制数转成 1 个 8 进制数 例如：100 14+02+01=4
	for(i=0;i<len-2;i=i+3){
		cout<<(s[i]-'0')4 + (s[i+1]-'0')2 + (s[i+2]-'0')1 ;
	}
	return 0;
}
```

二进制转十六进制

思路：

第一步：判断字符串的长度是否是 4 的倍数，如果不是，则补 0。

第二步：每 4 位 2 进制转换为 1 位的 16 进制输出。

```
#include <bits/stdc++.h>//2-1294  javacn
using namespace std;

/*
1. 判断字符串长度是否是 4 的倍数，如果不是补 0
2. 将每 4 位的 2 进制转换为 1 位的 16 进制，输出
*/
string s;// 存放 2 进制
string t = "0123456789ABCDEF";
int main(){
	cin>>s;
	// 如果长度不是 4 的倍数，补 0
	//11010
	if(s.size()%4==1) s = "000" + s;
	else if(s.size()%4==2) s = "00" + s;
	else if(s.size()%4==3) s = "0" + s;

	//cout<<s<<endl;
	// 计算
	//0101,1010
	for(int i = 0;i < s.size();i = i + 4){
		// 将 4 位的 2 进制：s[i] s[i+1] s[i+2] s[i+3]
		// 转 16 进制
		int x = (s[i+3]-'0')*1+(s[i+2]-'0')*2+(s[i+1]-'0')*4+(s[i]-'0')*8;
		cout<<t[x];
	}
	return 0;
}
```

/*1359 - 【基础】八进制转换二进制

题目描述　请将一个 100 位以内的 8 进制整数转换为 2 进制整数！

输入　100 位以内的 8 进制整数　　输出　该数对应的 2 进制整数

样例

输入 12376532347173217361

输出 1010011111110101011010011100111001111011010001111011110001

*/

八进制转换二进制

```
#include <bits/stdc++.h>//3-1359-1    xingdian119
using namespace std;
long long n2shi(string s, int jin) {  // 其他进制数转为 10 进制
  long long r = 0, t = 1;             // t: 表示权重
  for (int i = s.size() - 1; i >= 0; i--) {
    r += (s[i] - '0') * t;
    t *= jin;
  }
  return r;
}
string shi2n(long long n, int jin) {  // 10 进制转为其他进制数
  string t = "0123456789ABCDEF", r = "";
  if (n == 0)
    r = '0';
  else {
    while (n != 0) {
      r = t[n % jin] + r;
      n /= jin;
    }
  }
  return r;
}
int main() {
  string s;
  cin >> s;
  cout << shi2n(n2shi(s, 8), 2);
}
```

将每位的 8 进制转换为 3 位的 2 进制：

```
#include<bits/stdc++.h>//3-1359-2  javacn
using namespace std;
int main() {
	string s,r;
	int x;
	// 定义字符串数组，枚举所有情况
	string t[8]= {"000","001","010","011","100","101","110","111"};
	cin>>s;
	// 调用函数，求每位 8 进制对应的二进制
	for(int i=0; i<s.size(); i++) {
		x=s[i]-'0';
		r=r+t[x];
	}
	// 删除字符串前面的 0
	while(r[0]=='0') {
		r.erase(0,1);
	}

	if(r == "") cout<<0;
	else cout<<r;
	return 0;
}
```

十六进制转二进制

```
#include <bits/stdc++.h>//4-1306-1 yhh133
#define ll long long
using namespace std;
ll change1(string &s,int x){
	ll sum=0,t=1;
	for(int i=s.size()-1;i>=0;i--){
		if(s[i]>='A'&&s[i]<='F'){
			sum+=(s[i]-'A'+10)*t;
		}
		else sum+=(s[i]-'0')*t;
		t*=x;
	}
	return sum;
}
string change2(ll n,int t){
	string s;
	while(n){
		if(n%t>=10) s+=(n%t-10+'A');
		else s+=n%t+'0';
		n/=t;
	}
	reverse(s.begin(),s.end());
	return s;
}
int main() {
	string s;
	cin>>s;
	ll num=change1(s,16);
	if(num==0) cout<<0;
	else cout<<change2(num,2);
	return 0;
}
```

每位转成 4 位二进制，拼接后删除前导零

```
#include<bits/stdc++.h>//4-1306-2  jiangyf70
using namespace std;
int main()
{
    string s;
    cin >> s;
    if(s == "0") //特判 0
    {
        cout << 0;
        return 0;
    }
    int a;
    string h, x;
    for(int i = 0; i < s.size(); i++)
    {
        h = "";
        if(s[i] <= '9') a = s[i] -'0';
        else a = s[i] - 'A' + 10;
        for(int j = 0; j < 4; j++) //转成 4 位二进制
        {
            h = char(a % 2 + '0') + h;
            a /= 2;
        }
        x += h;
    }
    while(x[0] == '0') x.erase(0, 1);
    cout << x;
}
```

麻烦的做法，一个一个转换

1、输入字符串，遍历每个字符 2、将每个字符转换成二进制字符串 3、拼接到新的字符串中输出（先删除前导 0）

```
#include <bits/stdc++.h>//4-1306-3    1989444607
using namespace std;
// 将字符转换为字符串
string cTos(char c){
	string res ="";
	int num=0;
	if(isdigit(c)){   num = c-'0';      }
	else{ num = c-'A'+10;   }
	while(num){
		int x = num%2;
		res=char(x+'0')+res;
		num/=2;
	}
	// 字符转换为二进制字符串，特别注意字符 0 的情况
	if(res.size()%4==1){    res="000"+res;    }
	else if(res.size()%4==2){     res = "00"+res;   }
		else if(res.size()%4==3){     res = "0"+res;    }
			else if(res.size()==0){ res = "0000";     }// 字符为 0 时
	return res;
}
int main(){
	string s,s2;
	cin>>s;
	for(int i=0;i<s.size();i++){  s2+=cTos(s[i]);   }
	while(s2[0]=='0'){      s2.erase(0,1);    }     // 删除前导 0
	// 输出，特别注意 0 的情况
	if(s2==""){ cout<<0;    }
	else{ cout<<s2;   }
	return 0;
}
```

思路：将每一位的 16 进制数，转换为 4 位的二进制数！

第一步：将每位 16 进制转换为 4 位的 2 进制，连接到字符串上！

第二步：删除前导 0，也就是要从第一个非 0 开始输出！

```
#include <bits/stdc++.h>//4-1306-4    javacn
using namespace std;
/*
1. 将每一位的 16 进制（0~15）转 2 进制（不足 4 位补 0）
2. 删除前导 0
1A
0001,1010
*/
string t[16] = {"0000","0001","0010","0011","0100","0101","0110","0111","1000","
1001","1010","1011","1100","1101","1110","1111"};
string s;
string r = "";// 存储 2 进制的结果

int main(){
	cin>>s;
	// 循环将每一位的 16 进制转换为 4 位的 2 进制
	for(int i = 0;i < s.size();i++){
		// 如果是字符 '0'~'9'
		if(isdigit(s[i])){	r = r + t[s[i]-'0'];		}
		else{ r = r + t[s[i]-55]; } // 说明是 'A'（65）~'F'，换成 10~15
	}
	// 删除前导 0
	// 当 r 的第 1 位是 0，删除
	while(r[0]=='0'){
		r.erase(0,1);
	}
	// 如果输入为 0，那么结果是 0000，会被上面的 while 删前导 0，都删掉
	if(r == "") cout<<0;
	else cout<<r;
}
```

十六进制转换

思路：借助二进制为中间转换值，先将 16 进制转换成 2 进制，再将 2 进制转换成 8 进制
因为 1 位 16 进制数对应 4 位 2 进制数，3 位 2 进制数对应 1 位 8 进制数。

```
#include<bits/stdc++.h>//5-1295  javacn
using namespace std;
//a 数组存储：1 位 16 进制数对应 4 位 2 进制数
string a[]= {"0000","0001","0010","0011","0100","0101","0110","0111","1000","100
1","1010","1011","1100","1101","1110","1111"};
int main() {
	string s,r="",x;
	cin>>s;
	// 十六进制 -> 二进制
	int i,t;
	for(i=0; i<s.size(); i++) {
		if(s[i]>='A' && s[i]<='F') {  t = s[i]-'A'+10;  }
		else {	t = s[i]-'0';	}
		//t 是对应的下标
		r = r + a[t];
	}
	// 将前导 0 去掉
	while(r[0]=='0') r.erase(0,1);
	// 二进制数 r -> 八进制
	// 前导补 0  3 位二进制数为 1 组
	while(r.size()%3!=0) {
		r =  '0' + r;
	}
	//100 -> 14+02+01=4
	for(i=0; i<r.size(); i=i+3) {
		cout<<(r[i]-'0')*4 + (r[i+1]-'0')*2 + (r[i+2]-'0')*1;
	}
	return 0;
}
```

高精度运算

高精度运算基础

高精度加法

```
#include<iostream>//1-1268-1    jiangyf70
using namespace std;
int a[250], b[250], c[250];
int main()
{
    string s1, s2;
    cin >> s1 >> s2;
    for(int i = s1.size() - 1; i >= 0; i--) a[s1.size()- 1 - i] = s1[i] - '0';
    for(int i = s2.size() - 1; i >= 0; i--) b[s2.size() -1 - i] = s2[i] - '0';
    int len;
    if(s1.size() > s2.size()) len = s1.size();
    else len = s2.size();
    int t = 0;
    for(int i = 0; i < len; i++)
    {
        t += a[i] +b[i];
        c[i] = t % 10;
        t /= 10;
    }
    if(t) c[len++] = t;
    for(int i = len - 1; i >= 0; i--) cout << c[i];
    return 0;
}
```

```
/* 1 读入两个整数数组 倒序。2 比较次数 为最长的 。3 对应位相加 如果大于 10 后一位进位，否则直接增加。4 反向输出 */
#include <bits/stdc++.h>//1-1268-2    David0724
using namespace std;
string a1, b1,c1;
int a[300], b[300], c[300];
int i ,j;
int main(){
    cin >> a1 >> b1;
    // 将字符串转化为整型数组
    int l1 = a1.size(), l2 = b1.size();
    for  (i = l1 - 1,j = 0; i >= 0; i--,j++){   a[j] = a1[i] - '0';   }
    for  (i = l2 - 1,j = 0; i >= 0; i--,j++){   b[j] = b1[i] - '0';    }
    // for(i = 0; i < b1.size();i++){   cout << b[i] ; }
    //c 的长度应该位 a b 最长
    int l ;
    if (l1 > l2){
        l = l1;
    }else{
        l = l2;
    }

    for(i = 0; i < l; i++){
        c[i] += a[i] + b[i];
        // 对于当前位进行判断是否进位
        if(c[i] >= 10){
           c[i] -= 10;
           c[i + 1]++;
        }
    }
    if (c[l] != 0 ){     l++;   }
    reverse(c,c + l);     // 反转数组
    for(i = 0; i < l;i++){    cout << c[i] ;    }

}
```

/* 关于进位问题的判断

1 位数加上 3 位数最多是几位数？

2 + 99 = 101

应该观察长度最多数组的 长度位 即为最后一位的下一位

有内容时 结果数组的长度就为最多长度 + 1 位

*/

```cpp
#include <bits/stdc++.h>//1-1268-3    javacn
using namespace std;
string s1,s2;// 高精度整数
int a[250],b[250],c[500];
int i,j,len;
int main(){
    // 用 string 读入高精度整数
    cin>>s1>>s2;
    // 将两个高精度数逆序放入 ab 两个整数数组中
    for(i=0;i<s1.size();i++){    a[i] = s1[s1.size()-1-i] - '0';   }
    for(i=0;i<s2.size();i++){   b[i] = s2[s2.size()-1-i] - '0';   }
    // 从左往右，逐位求和，结果存入 c 数组
    // 加法的次数取决于两个整数中较长的字符串
    len = s1.size();
    if(s2.size()>s1.size()){
        len = s2.size();
    }
    // 逐位相加
    for(i=0;i<len;i++){   c[i] = a[i]+b[i];    }
    // 逐位进位
    for(i=0;i<len;i++){
        if(c[i]>=10){
            c[i+1] = c[i+1] + c[i] / 10;
            c[i] = c[i] % 10;
        }
    }
    // 逆序输出结果
    // 两个不超过 len 位的整数做加法，结果可能是 len+1 位
    if(c[len]!=0){
        len++;
    }
    for(i=len-1;i>=0;i--){    cout<<c[i];    }
    return 0;
}
```

高精度减法

```
#include<iostream>//2-1269-1  jiangyf70
using namespace std;
int a[250], b[250], c[250];
string sub(string s1, string s2)
{
    char f = '+';// 一定要赋值，我在这卡了很久
    if(s1.size() < s2.size() || s1.size() == s2.size() && s1 <  s2)
    {
       swap(s1, s2);
       f = '-';
    }
    for(int i = 0; i < s1.size(); i++) a[s1.size() - 1 - i] = s1[i] - '0';
    for(int i = 0; i < s2.size(); i++) b[s2.size() - 1 - i] = s2[i] - '0';
    int l = s1.size();
    for(int i = 0; i < l; i++)
    {
        if(a[i] < b[i])  {     a[i+1]--;     a[i] += 10;   }
        c[i] = a[i] - b[i];
    }

    while(c[l] == 0 && l >= 0) l--;
    string s3 = "";
    for(int i = 0; i <= l; i++)     {     s3 = char(c[i] + '0') + s3;   }
    if(f ==  '-' ) s3 = f + s3;
    if(l == -1) s3 = "0";
    return s3;
}
int main()
{
    string a, b;
    cin >> a >> b;
    cout << sub(a, b);
    return 0;
}
```

第一步：判断正负，如果 s1 比 s2 对应的整数小，结果为负，交换 s1 s2

第二步：将两个字符串，逆序存入 2 个整数输出测试

第三步：从左至右，逐位相减，不够借位 第四步：从右向左，逆序输出

```
#include <bits/stdc++.h>//2-1269-2  javacn
using namespace std;
/*
   高精度减法：
   第一步：判断正负，如果 s1 比 s2 对应的整数小，结果为负，交换 s1 s2
   第二步：将两个字符串，逆序存入 2 个整数输出测试
   第三步：从左至右，逐位相减，不够借位
   第四步：从右向左，逆序输出
*/
string s1,s2;
int a[250],b[250],c[250];
int i,len,p;
char f = '+';// 表示结果的正负

int main(){
	cin>>s1>>s2;
	// 长的一定大，一样长字典码大的一定大
	//"123" "3"   "123" "125"
	if(s1.size() < s2.size() || (s1.size() == s2.size() && s1 < s2)){
		f = '-';
		swap(s1,s2);// 直接交换两个变量的值
	}

	//cout<<f<<" "<<s1<<" "<<s2;

	// 将 s1 和 s2 逆序存入整数数组
	for(i = 0;i < s1.size();i++){
		//0 -> s1[s1.size()-1]
		//1 -> s1[s1.size()-2]
		a[i] = s1[s1.size() - i - 1] - '0';
	}
```

```
    for(i = 0;i < s2.size();i++){
        b[i] = s2[s2.size() - i - 1] - '0';
    }

    // 逐位相减
    len = s1.size();


    for(i = 0;i < len;i++){
        // 如果不够减，向右借 1，当 10 用
        if(a[i] < b[i]){
            a[i + 1] = a[i + 1] - 1;
            a[i] = a[i] + 10;
        }

        c[i] = a[i] - b[i];
    }

    // 判断是否要输出负号
    if(f == '-') cout<<f;

    // 从右向左逐位输出，从第一个遇到的非 0 元素开始输出
    for(i = len - 1;i >= 0;i--){
        if(c[i] != 0){
            p = i;
            break;
        }
    }

    // 逆序从第一个非 0 元素 输出每一位
    for(i = p;i >= 0;i--){
        cout<<c[i];
    }

}
```

高精度乘单精度

1. 将高精度整数 s1 逆序存储。2. 将 a 数组逐位乘 b，结果存入 c 数组。3. 进位。

4. 逆序从第 1 个非 0 开始打印，要注意多考虑 4 位。

```
#include<bits/stdc++.h>//3-1286-1   javacn
using namespace std;
string s1;
int a[210],b,c[210];
int main() {
	cin>>s1>>b;
	//1. 逆序存储
	for(int i = 0;i < s1.size();i++){
		a[i] = s1[s1.size()-i-1] - '0';
	}
	//2. 逐位相乘
	for(int i = 0;i < s1.size();i++){
		c[i] = a[i] * b;
	}
	//3. 逐位进位
	for(int i = 0;i < s1.size() + 4;i++){
		if(c[i] >= 10){
			c[i+1] = c[i+1] + c[i] / 10;
			c[i] = c[i] % 10;
		}
	}
	//4. 逆序从第 1 个非 0 开始打印
	int p = 0;
	for(int i = s1.size() + 4 - 1;i >= 0;i--){
		if(c[i] != 0){
			p = i;
			break;
		}
	}
	for(int i = p;i >= 0;i--){    cout<<c[i]; }
	return 0;
}
```

高精度乘单精度

```
#include<bits/stdc++.h>//3-1286-2  jiangyf70
using namespace std;
string multi(string s1, string s2)
{
    string s3;
    int a[250], b[250], c[250];
    for(int i = 0; i < 250; i++)   {     a[i] = 0, b[i] = 0, c[i] = 0;    }
    if(s1.size() < s2.size()) swap(s1, s2);
    int x = stoi(s2);
    int len = s1.size();
    for(int i = 0; i < len; i++)  a[len - 1 - i] = s1[i] -  '0' ;
    int t = 0, i;
    for(i = 0; i < len || t; i++)
    {
            t = a[i] * x + t;
            c[i] = t % 10;
            t /= 10;
    }
    len = i;
    for(i = len - 1; i >= 0; i--)
    {
        s3 =  s3 + char(c[i] +  '0' );
    }
    if(x == 0) return "0";
    return s3;
}
int main()
{
    string a, b;
    cin >> a >> b;
    cout << multi(a, b);
    return 0;
}
```

高精度乘

模拟乘法的数学计算过程，逐位相乘，错位相加。

```
#include <bits/stdc++.h>//4-1287  javacn
using namespace std;
string s1,s2;
int a[250],b[250],c[500];
int i,j,p;

int main(){
	cin>>s1>>s2;
	// 逆序将 s1 和 s2 存入 ab 数组
	for(i = 0;i < s1.size();i++){ a[i] = s1[s1.size() - i - 1] - '0';	}
	for(i = 0;i < s2.size();i++){ b[i] = s2[s2.size() - i - 1] - '0';	}
	// 循环 a 数组的每一位，用 a[i] 去乘以 b 数组的每一位 b[j]
	// 结果错位加到 c 数组的 c[i+j] 这一位上
	for(i = 0;i < s1.size();i++){
		for(j = 0;j < s2.size();j++){
			c[i+j] = c[i+j] + a[i] * b[j];
			// 进位
			if(c[i+j] >= 10){
				c[i+j+1] = c[i+j+1] + c[i+j] / 10;
				c[i+j] = c[i+j] % 10;
			}
		}
	}
	// 逆序输出，逆序从第一个非 0 元素位置开始输出
	for(i = s1.size() + s2.size() - 1;i >= 0;i--){
		if(c[i] != 0){	p = i;	break;	}
	}

	for(i = p;i >= 0;i--){
		cout<<c[i];
	}
	return 0;
}
```

高精度整数除法

```
#include<bits/stdc++.h>//5-1271-1 jiangyf70
using namespace std;
string div(int a, int b, int n)
{
    string s = to_string(a / b) + '.';
    int t = a % b;
    for(int i = 0; i < n; i++)
    {
        t *= 10;
        s += char(t / b + '0');
        t %= b;
    }
    return s;
}

int main()
{
    int a, b, n;
    cin >> a >> b >> n;
    cout << div(a, b, n);
    return 0;
}
```

```cpp
#include<bits/stdc++.h>//5-1271-2  javacn
using namespace std;

int main(){
    int a,b,n,t,i;
    cin>>a>>b>>n;
    //a/b 为整数部分
    cout<<a/b<<".";
    // 例如 97/61 整数部分为 1 接下来就是小数部分
// 先求得余数 36
    t = a % b;
    for(i=1;i<=n;i++){
    // 余数 *10
        t = t * 10;
        // 小数部分
        cout<<t/b;
        t = t % b;
    }
    return 0;
}
```

求 2 的 n 次方

准备一个整数数组 a ，存放 2 的 n 次方， a 数组默认存储一个 1 ，代表 2 的 0 次方！ 循环 n 次，每次循环都要将 a 数组的每一位 ×2，并进位，然后判断 a 数组 ×2 后是否多出一位，如果多出一位，a 数组位数计数器 k++。

逆序输出 a 数组的 k 个数。

```
#include <bits/stdc++.h>//6-1280     javacn
using namespace std;
int a[100] = {1};
//k 代表 a 数组元素的个数，代表了高精度的整数的位数
int i,j,k = 1,n;
int main(){
	cin>>n;
	// 循环 n 次，每次都将 a 数组 * 2
	for(i = 1;i <= n;i++){
		// 将 a 数组的每一位都 * 2
		for(j = 0;j < k;j++){   a[j] = a[j] * 2;  }
		// 逐位进位
		for(j = 0;j < k;j++){
			if(a[j] >= 10){
				a[j+1]=a[j+1]+a[j]/10;
				a[j]=a[j]%10;
			}
		}

		// 判断 a 数组是否多出一位
		if(a[k] != 0){    k++;  }
	}
	// 逆序输出 a 数组的 k 个数
	for(i = k - 1;i >= 0;i--){
		cout<<a[i];
	}
	return 0;
}
```

求 2+2*2+2*2*2+…+2*2*2*….*2

```
#include<bits/stdc++.h>//7-1281-1    w2016010182
using namespace std;
string s1;
int a1[241],b1[245];
int n;
string f;
string r;
string ss(string s1,int n){
      int len;
      string s3="";
      for(int i=0;i<s1.size();i++)  {a1[i]=s1[s1.size()-i-1]-'0'; }
      len=s1.size();
      for(int i=0;i<len;i++)  {     b1[i]=a1[i]*n;    }
      for(int i=0;i<len;i++)
      {
          b1[i+1]+=b1[i]/10;
            b1[i]%=10;
            if(b1[len]>0)     {     len++;      }
      }
      while(b1[len]==0&&len>=1)     {     len--;      }
      for(int i=len;i>=0;i--)       {     s3=s3+char(b1[i]+'0');  }
      return s3;
}
```

```
int a[250],b[250],c[250];
string st(string s1,string s2)
{
	int len;
	for(int i=0;i<s1.size();i++)  {    a[s1.size()-i-1]=s1[i]-'0';   }
	for(int i=0;i<s2.size();i++)  {    b[s2.size()-i-1]=s2[i]-'0';   }
	len=s1.size();
	if(s1.size()<s2.size()) {    len=s2.size();    }
	for(int i=0;i<len;i++)  {    c[i]=a[i]+b[i];   }
	for(int i=0;i<len;i++)
	{
		c[i+1]=c[i+1]+c[i]/10;
		c[i]=c[i]%10;
	}
	while(c[len]==0&&len>=1)      {    len--;      }
	string s3="";
	for(int i=len;i>=0;i--) {    s3=s3+char(c[i]+'0');   }
	return s3;
}
int main()
{
	cin>>n;
	r="0";
	f="1";
	for(int i=1;i<=n;i++)
	{
	    f=ss(f,2);
	    r=st(f,r);
	}
	cout<<r;
}
```

标准化高精度乘法和加法函数

```
#include<iostream>//7-1281-2   anselxu
#include<cstring>
#include<algorithm>
#include<cmath>
//#define unsigned long long LL
using namespace std;
int a[255],b[255],c[255],sum[255];
int zh(string s,int *arr){
	int len=s.length();
	for(int i=1;i<=len;i++)
		arr[i]=s[len-i]-48;
	return len;
}
// 乘法
void gs(int a[],int b[],int c[]){
	for(int i=1;i<=a[0];i++){
		int jw=0;
		for(int j=1;j<=b[0];j++){
			c[i+j-1]+=a[i]*b[j]+jw;
			jw=c[i+j-1]/10;
			c[i+j-1]%=10;
		}
		if(jw) c[i+b[0]]+=jw;
	}
	c[0]=a[0]+b[0];
	while(c[c[0]]==0&&c[0]>1) c[0]--;
}
// 加法
```

```
// 加法
void gj(int a[],int b[],int c[]){
	int jw=0,i=1;
	while(i<=a[0]||i<=b[0]){
		c[i]=a[i]+b[i]+jw;
		jw=c[i]/10;
		c[i]%=10;
		i++;
	}
	c[0]=i-1;
	if(jw) c[++c[0]]=jw;
	return;
}
int main(){
	int n;
	cin>>n;
	for(int j=1;j<=n;j++){
		// 每次计算 2 的 j 次方，初始化两个乘数
		b[0]=a[0]=a[1]=1;
		b[1]=2;
		for(int i=1;i<=j;i++){
			//a 数组累计乘 2，每次计算用 c 存储积，利用 memcpy 拷贝到 a，运算
前清空 c
			memset(c,0,sizeof(c));
			gs(a,b,c);
			memcpy(a,c,sizeof(c));
		}
		gj(sum,c,sum);// 累加
	}
	for(int i=sum[0];i>0;i--)
		cout<<sum[i];
	return 0;
}
```

写成函数更清晰

```
#include<bits/stdc++.h>//7-1281-3    jiangyf70
using namespace std;

string mul(string s, int x)
{
    int a[100], b[100];
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));
    for(int i = s.size() - 1; i >= 0; i--) a[s.size() - 1 - i] = s[i] - '0';
    int len = s.size();
    for(int i = 0; i < len; i++)
    {
        b[i] = a[i] * x;
    }
    for(int i = 0; i < len; i++)
    {
        b[i + 1] += b[i] / 10;
        b[i] %= 10;
    }
    if(b[len])
    {
        b[len + 1] = b[len] / 10;
        b[len] %= 10;
        len++;
    }

    string s1;
    for(int i = len - 1; i >= 0; i--)
        s1 += b[i] + '0';
    return s1;

}
```

```
string add(string s1, string s2)
{
    int a[100], b[100], c[100];
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));
    if(s1.size() < s2.size()) swap(s1, s2);
    for(int i = s1.size() - 1; i >= 0; i--) a[s1.size() - 1 - i] = s1[i] - '0';
    for(int i = s2.size() - 1; i >= 0; i--) b[s2.size() - 1 - i] = s2[i] - '0';
    int len = s1.size();
    for(int i = 0; i < len; i++)   {     c[i] = a[i] + b[i];   }
    for(int i = 0; i < len; i++)   {    c[i+1] += c[i] / 10;   c[i] %= 10;   }
    if(c[len])
    {
        c[len + 1] = c[len] / 10;
        c[len] %= 10;
        len++;
    }
    string s3 = "";
    for(int i = len - 1; i >= 0; i--)    {     s3 = s3 + char(c[i] + '0');   }
    return s3;
}
int main()
{
    int n;
    cin >> n;
    string s = "2";
    string sum = "2";
    for(int i = 2; i <= n; i++)
    {
        s = mul(s, 2);
        sum = add(sum, s);
    }
    cout << sum;
    return 0;
}
```

高精度 * 整数 以及 高精度求和的结合。

```
#include <bits/stdc++.h>//7-1281-4    javacn
using namespace std;
int a[100] = {1};
int r[1000];
//k 代表 a 数组元素的个数，代表了 2 的 n 次方高精度的整数的位数
//k2 代表了高精度的总和的位数
int i,j,k = 1,n,k2 = 1,len;
int main() {
	cin>>n;
	// 循环 n 次，每次都将 a 数组 * 2
	for(i = 1; i <= n; i++) {
		// 将 a 数组的每一位都 * 2
		for(j = 0; j < k; j++) {
			a[j] = a[j] * 2;
		}
		// 逐位进位
		for(j = 0; j < k; j++) {
			if(a[j] >= 10) {
				a[j+1]=a[j+1]+a[j]/10;
				a[j]=a[j]%10;
			}
		}
		// 判断 a 数组是否多出一位
		if(a[k] != 0) {
			k++;
		}
		// 求出了 2 的 i 次方，结果为 k 位
		// 将 k 位的 2 的 i 次方，加到 k2 位的总和 r 上
```

```
            //求出了2的i次方，结果为k位
            //将k位的2的i次方，加到k2位的总和r上
            len = k;
            if(k2 > k) len = k2;
            for(j = 0; j < len; j++) {
                  r[j] = r[j] + a[j];
                  //进位
                  if(r[j] >= 10) {
                        r[j+1]=r[j+1]+r[j]/10;
                        r[j]=r[j]%10;
                  }
            }
            //判断r数组是否多了1位
            if(r[k2]!=0) {
                  k2++;
            }
      }

      //输出r数组的结果
      for(i = k2 - 1; i >= 0; i--) {
            cout<<r[i];
      }
      return 0;
}
```

计算 N 的阶乘

高精度乘单精度，注意结果可能会多进 2 位：

```
#include <bits/stdc++.h>//8-1285  javacn
using namespace std;
int a[200] = {1},i,n,j,len = 1;
int main(){
    cin>>n;
    for(i = 1;i <= n;i++){
        // 高精度的 a 乘以整数 i
        // 逐位相乘
        for(j = 0;j < len;j++){
            a[j] = a[j] * i;
        }

        // 逐位进位，由于 i 可能是 2 位数，可能多进 2 位
        for(j = 0;j < len + 2;j++){
            if(a[j] >= 10){
                a[j + 1] = a[j + 1] + a[j] / 10;
                a[j] = a[j] % 10;
            }
        }

        // 判断是否多进 2 位或者 1 位
        if(a[len+1] != 0) len = len + 2;
        else if(a[len] != 0) len = len + 1;
    }

    // 逆序输出
    for(i = len - 1;i >= 0;i--){
        cout<<a[i];
    }
      return 0;
}
```

求 1!+2!+3!+4!+...+n!

高精度 * 单精度以及高精度求和问题。

```
#include<bits/stdc++.h>//9-1296-1  javacn
using namespace std;
int a[1000] = {1};// 表示阶乘
int r[1000];// 表示总和
int k,len;//k 表示 a 数组下标，len 表示 r 数组下标
int n;
int main() {
    cin>>n;
    for(int i = 1;i <= n;i++){
        // 将 i 乘到阶乘上（a 数组）
        for(int j = 0;j < k;j++){    a[j] = a[j] * i;     }
        for(int j = 0;j < k + 2;j++){        // 进位
            if(a[j] >= 10){
                a[j+1] = a[j+1] + a[j] / 10;
                a[j] = a[j] % 10;
            }
        }
        if(a[k+1] != 0) k = k + 2;        // 判断结果是否多 2 位或者 1 位
        else if(a[k] != 0) k++;
        len = max(k,len);        // 求和
        for(int j = 0;j < len;j++){    r[j] = r[j] + a[j];     }
        for(int j = 0;j < len;j++){        // 进位
            if(r[j] >= 10){
                r[j+1] = r[j+1] + r[j] / 10;
                r[j] = r[j] % 10;
            }
        }
        if(r[len] != 0) len++;
    }
    for(int i = len - 1;i >= 0;i--){    cout<<r[i];    }      // 输出结果
    return 0;
}
```

```
#include<bits/stdc++.h>//9-1296-2    w2016010182
using namespace std;
int n;
string x,y,s,h=" 0" ;
int a[25000],b[25000],c[50000],p;
string gjc(string s1,string s2)
{
      memset(a,0,sizeof(a));
      memset(b,0,sizeof(b));
      memset(c,0,sizeof(c));
      for(int i=0;i<s1.size();i++)   {     a[i]=s1[s1.size()-i-1]-'0';    }
      for(int i=0;i<s2.size();i++)   {     b[i]=s2[s2.size()-i-1]-'0';    }
      for(int j=0;j<s2.size();j++)
      {
            for(int i=0;i<s1.size();i++)
            {
                  c[i+j]=a[i]*b[j]+c[i+j];
                  if(c[i+j]>=10)
                  {
                        c[i+j+1]=c[i+j+1]+c[i+j]/10;
                        c[i+j]=c[i+j]%10;
                  }
            }
      }
      int len=s1.size()+s2.size();
      while(c[len]==0&&len>1) {     len--;      }
      string s3=" " ;
      for(int i=len;i>=0;i--) {     s3=s3+char(c[i]+' 0' ); }
      return s3;
}
```

```
string gjj(string s1,string s2)
{
	memset(a,0,sizeof(a));
	memset(b,0,sizeof(b));
	memset(c,0,sizeof(c));
	for(int i=0;i<s1.size();i++)  {     a[s1.size()-i-1]=s1[i]-'0';   }
	for(int i=0;i<s2.size();i++)  {     b[s2.size()-i-1]=s2[i]-'0';   }
	int l=s1.size();
	if(s1.size()<s2.size()) {     l=s2.size();      }
	for(int i=0;i<l;i++)    {     c[i]=a[i]+b[i];         }
	for(int i=0;i<l;i++)
	{
		if(c[i]>=10)
		{
			c[i+1]=c[i+1]+c[i]/10;
			c[i]=c[i]%10;
		}
	}
	string s3="";
	if(c[l]==0&&l>1)  {     l--;  }
	for(int i=l;i>=0;i--)   {     s3=s3+char(c[i]+'0');   }
	return s3;
}
int main()
{
	cin>>n;
	y="1";
	x="1";
	s="1";
	h="0";
	for(int i=1;i<=n;i++)
	{     x=gjc(x,y); h=gjj(h,x); y=gjj(y,s); }
	cout<<h;
	return 0;
}
```

```
#include<bits/stdc++.h>//9-1296-3    w2016010182
using namespace std;
string qz;
string a1;
int q1[241],q3[245];
int a[250],b[250],c[250];
int n;
string gjc(string q,int m){
      int i,e;
      string s3="";
      memset(q1,0,sizeof(q1));
      memset(q3,0,sizeof(q3));
      for(i=0;i<q.size();i++)
      {
            q1[i]=q[q.size()-i-1]-'0';
      }
      e=q.size();
      for(i=0;i<e;i++)
      {
            q3[i]=q1[i]*m;
      }
      for(i=0;i<e;i++)
      {
          q3[i+1]+=q3[i]/10;
            q3[i]%=10;
            if(q3[e]>0)
            {
                  e++;
            }
      }
      while(q3[e]==0&&e>=1)    {     e--;  }
      for(i=e;i>=0;i--) {     s3=s3+char(q3[i]+'0');  }
      return s3;
}
```

```
string gjj(string s1,string s2)
{
	int len;
	memset(a,0,sizeof(a));
	memset(b,0,sizeof(b));
	memset(c,0,sizeof(c));
	for(int i=0;i<s1.size();i++)  {  a[s1.size()-i-1]=s1[i]-'0';  }
	for(int i=0;i<s2.size();i++)  {  b[s2.size()-i-1]=s2[i]-'0';  }
	len=s1.size();
	if(s1.size()<s2.size()) {  len=s2.size();  }
	for(int i=0;i<len;i++)  {  c[i]=a[i]+b[i];  }
	for(int i=0;i<len;i++)
	{
		c[i+1]=c[i+1]+c[i]/10;
		c[i]=c[i]%10;
	}
	while(c[len]==0&&len>=1)  {  len--;  }
	string s3="";
	for(int i=len;i>=0;i--) {  s3=s3+char(c[i]+'0');  }
	return s3;
}
int main()
{
	cin>>n;
	a1="0";
		for(int i=1;i<=n;i++)
		{
			qz="1";
			for(int j=1;j<=i;j++)  {  qz=gjc(qz,j);  }
			a1=gjj(a1,qz);
		}
		cout<<a1<<endl;
	return 0;
}
```

```
#include<bits/stdc++.h>//9-1296-4    w2016010182
using namespace std;
string qz;
string a1;
int q1[241],q3[245];
int a[250],b[250],c[250];
int n;
string gjc(string q,int m){
	int i,e;
	string s3="";
	for(i=0;i<q.size();i++)
	{
		q1[i]=q[q.size()-i-1]-'0';
	}
	e=q.size();
	for(i=0;i<e;i++)
	{
		q3[i]=q1[i]*m;
	}
	for(i=0;i<e;i++)
	{
	    q3[i+1]+=q3[i]/10;
		q3[i]%=10;
		if(q3[e]>0)
		{
			e++;
		}
	}
	while(q3[e]==0&&e>=1)
	{
		e--;
	}
	for(i=e;i>=0;i--) {	s3=s3+char(q3[i]+'0');  }
	return s3;
}
```

```
string gjj(string s1,string s2)
{
	int len;
	for(int i=0;i<s1.size();i++)  {	a[s1.size()-i-1]=s1[i]-'0';  }
	for(int i=0;i<s2.size();i++)  {	b[s2.size()-i-1]=s2[i]-'0';  }
	len=s1.size();
	if(s1.size()<s2.size()) {	len=s2.size();  }
	for(int i=0;i<len;i++)  {	c[i]=a[i]+b[i];  }
	for(int i=0;i<len;i++)
	{
		c[i+1]=c[i+1]+c[i]/10;
		c[i]=c[i]%10;
	}
	while(c[len]==0&&len>=1)	{	len--;	}
	string s3="";
	for(int i=len;i>=0;i--) {	s3=s3+char(c[i]+'0');  }
	return s3;
}
int main()
{
	cin>>n;
	a1="0";
	 qz="1";
		for(int i=1;i<=n;i++)
		{
			qz=gjc(qz,i);
			a1=gjj(a1,qz);
		}
		cout<<a1<<endl;
	return 0;
}
```

高精度乘法和加法

```
#include<iostream>//9-1296-5    anselxu
#include<cstring>
#include<algorithm>
#include<cmath>
//#define unsigned long long LL
using namespace std;
int a[255],b[255],c[255],sum[255];
// 高精度乘法，a*b=c
void gs(int a[],int b[],int c[]){
	for(int i=1;i<=a[0];i++){
		int jw=0;
		for(int j=1;j<=b[0];j++){
			c[i+j-1]+=a[i]*b[j]+jw;
			jw=c[i+j-1]/10;
			c[i+j-1]%=10;
		}
		if(jw) c[i+b[0]]+=jw;
	}
	c[0]=a[0]+b[0];
	while(c[c[0]]==0&&c[0]>1) c[0]--;
}
// 高精度加法 a+b=c
void gj(int a[],int b[],int c[]){
	int jw=0,i=1;
	while(i<=a[0]||i<=b[0]){
		c[i]=a[i]+b[i]+jw;
		jw=c[i]/10;
		c[i]%=10;
		i++;
	}
	c[0]=i-1;
	if(jw) c[++c[0]]=jw;
	return;
}
```

```
int main(){
	int n;
	cin>>n;

	for(int j=1;j<=n;j++){
		a[0]=a[1]=1;// 累乘初始值设 1
		for(int i=1;i<=j;i++){
			memset(c,0,sizeof(c));
			memset(b,0,sizeof(b));
			int x=i;
			while(x){
				b[++b[0]]=x%10;
				x/=10;
			}
			gs(a,b,c);// 累计乘法，所以每次将结果拷贝给下一次的乘数
			memcpy(a,c,sizeof(c));
		}
		gj(sum,c,sum); 求和
	}

	for(int i=sum[0];i>0;i--)
		cout<<sum[i];
	return 0;
}
```

棋盘里的麦子

求 2 的 n-1 次方的高精度计算的结果。

```
#include <iostream>//10-1409-1  javacn
using namespace std;
int main(){
    int a[110] = {1},n,i,k = 1,j;
    cin>>n;
    for(i = 1;i < n;i++){
        for(j = 0;j < k;j++){
            a[j] = a[j] * 2;
        }

        for(j = 0;j < k;j++){
            if(a[j] >= 10){
                a[j + 1] += a[j] / 10;
                a[j] = a[j] % 10;
            }
        }

        if(a[k] != 0){
            k++;
        }
    }

    for(i = k - 1;i >= 0;i--){
        cout<<a[i];
    }
    return 0;
}
```

```cpp
#include<bits/stdc++.h>//10-1409-2    w2016010182
using namespace std;
string s1;
int a[241],b[245];
int n;
string ss(string s1,int n){
	int len;
	string s3="";
	for(int i=0;i<s1.size();i++)  {     a[i]=s1[s1.size()-i-1]-'0';   }
	len=s1.size();
	for(int i=0;i<len;i++)  {     b[i]=a[i]*n;      }
	for(int i=0;i<len;i++)
	{
	    b[i+1]+=b[i]/10;
	      b[i]%=10;
	      if(b[len]>0)      {     len++;      }
	}
	while(b[len]==0&&len>=1)      {     len--;      }
	for(int i=len;i>=0;i--)  {     s3=s3+char(b[i]+'0');   }
	return s3;
}
int main(){
	cin>>n;
	s1=ss("1",1);
	for(int i=2;i<=n;i++)
	{
	      s1=ss(s1,2);
	}
	cout<<s1;
	return 0;
}
```

高精度运算应用

```
#include<bits/stdc++.h>//1-1279-1      javacn
using namespace std;
string s1,s2;
string opt;// 操作
string r = "";// 存储计算结果
int main()
{
      cin>>s1>>s2>>opt;
      if(s1.size()<s2.size()) swap(s1,s2);
    // 在 s2 前面补 0
      int len = s1.size() - s2.size();
      for(int i = 1;i <= len;i++) s2 = "0" + s2;
    // 逐位计算
      for(int i = 0;i < s1.size();i++){
            if(opt == "and"){
                  if(s1[i]=='1'&&s2[i]=='1') r = r + '1';
                  else r = r + '0';
            }
            else if(opt == "or"){
                  if(s1[i]=='1'||s2[i]=='1') r = r + '1';
                  else r = r + '0';
            }
                  else{
                        if(s1[i]!=s2[i]) r = r + '1';
                        else r = r + '0';
                        }
      }
      while(r[0] == '0') r.erase(0,1);     // 删除前导 0
      if(r == "") cout<<0;
      else cout<<r;
      return 0;
}
```

小 X 与位运算 (bignum)

解法一：

1. 先在短的字符串前面补 0，使得两个字符串一样长
2. 逐位计算
3. 删除前导 0，输出，注意判断结果计算结果为 0 的情况（这种情况下结果会被删成空字符串）

```cpp
#include <iostream>//1-1279-2     javacn
using namespace std;
string s1,s2,t;
int a[100010],b[100010],c[100010];
int main(){
      cin>>s1>>s2>>t;
    //1. 逆序存储
      for(int i = 0;i < s1.size();i++){
            a[i] = s1[s1.size()-i-1] - '0';
      }
      for(int i = 0;i < s2.size();i++){
            b[i] = s2[s2.size()-i-1] - '0';
      }
      int len = max(s1.size(),s2.size());     //2. 逐位运算
      for(int i = 0;i < len;i++){
            if(t == "and"){
                  if(a[i]==1&&b[i]==1) c[i]=1;
                  else c[i] = 0;
            }
            else if(t == "or"){
                  if(a[i]==0&&b[i]==0) c[i]=0;
                  else c[i] = 1;
            }
                  else if(t == "xor"){
                        if(a[i]!=b[i]) c[i]=1;
                        else c[i]=0;
                  }
      }
      int p = 0;     //3. 逆序打印
      for(int i = len - 1;i >= 0;i--){
            if(c[i] != 0){   p = i;     break;     }
      }
      for(int i = p;i >=  0;i--){  cout<<c[i];   }
      return 0;
}
```

解法二：

1. 将两个字符串逆序存储
2. 按照题意，分别判断如果计算方式是 and、or、xor，逐位计算结果
3. 逆序从第 1 个非 0 元素开始打印

```
#include <bits/stdc++.h>//2-1369-1    javacn
using namespace std;
string he(string s1,string s2){
	string r;// 存放总和
	int a[1000] = {0};
	int b[1000] = {0};
	int c[1000] = {0};
	// 逆序存入整数数组
	int i;
	for(i = 0;i < s1.size();i++){
		a[i] = s1[s1.size() - i - 1] -'0';
	}
	for(i = 0;i < s2.size();i++){
		b[i] = s2[s2.size() - i - 1] - '0';
	}
	// 逐位相加，逐位进位
	int len = s1.size();
	if(s2.size() > s1.size()) len = s2.size();

	for(i = 0;i < len;i++){
		c[i] = c[i] + a[i] + b[i];
		if(c[i] >= 10){
			c[i+1] = c[i+1] + c[i] / 10;
			c[i] = c[i] % 10;
		}
	}
	// 判断是否多出 1 位
	if(c[len] != 0) len++;
	// 逆序将 c 数组拼接乘字符串
	for(i = len-1;i>=0;i--){
		// 将 c[i]+'0' 这个 ascii 码强制转换为 char 类型
		r = r + (char)(c[i] +  '0' );
	}
	return r;
}
```

Pell 数列

解法一：将高精度求和、高精度 * 单精度定义为函数（其实高精度 * 单精度的函数可以没有，因为一个数 *2，可以转换为：该数 + 该数，参考解法二）。

```
/*
  An = A(n-1)*2 + A(n-2)
*/
// 求两个高精度的整数的和
```

```
// 求一个高精度的整数 * 2 的积
string cheng(string s){
	string r;
	int a[1000] = {0};
	int i;
	// 逆序存入 a 数组
	for(i = 0;i < s.size();i++){  a[i] = s[s.size() - i - 1] - '0';   }
	// 逐位 *2
	for(i = 0;i < s.size();i++){  a[i] = a[i] * 2;  }
	// 逐位进位
	for(i = 0;i < s.size();i++){
		if(a[i] >= 10){
			a[i+1] = a[i+1] + a[i] / 10;
			a[i] = a[i] % 10;
		}
	}
	// 判断是否多一位
	int len = s.size();
	if(a[len] != 0) len++;
	// 逆序拼接到字符串 r 上
	for(i = len - 1;i >= 0;i--){  r = r + to_string(a[i]);      }
	return r;
}
```

```
int main(){
	//z：代表计算结果，xy 代表 z 的前两项
	string x,y,z;
	int i,n;
	cin>>n;
	//A(n)=A(n-1)*2+A(n-2)
	x = "1";
	y = "2";
	if(n == 1){cout<<x;	}
	else if(n == 2){cout<<y;	}
		else{
		// 从第 3 项开始递推
				for(i = 3;i <= n;i++){
				z = he(cheng(y),x);
			// 修改 xy 的值，逐步向后推导
				x = y;
				y = z;
				}
		cout<<z;
		}
}
```

解法二：定义高精度求和函数，将高精度 *2，转换为高精度 * 高精度。

```
#include <bits/stdc++.h>//2-1369-2    javacn
using namespace std;
/*
  An = A(n-1)*2 + A(n-2)
*/
// 求两个高精度的整数的和
string he(string s1,string s2){
    string r;// 存放总和
    int a[1000] = {0};
    int b[1000] = {0};
    int c[1000] = {0};

    // 逆序存入整数数组
    int i;
    for(i = 0;i < s1.size();i++){
        a[i] = s1[s1.size() - i - 1] - '0';
    }

    for(i = 0;i < s2.size();i++){
        b[i] = s2[s2.size() - i - 1] - '0';
    }

    // 逐位相加，逐位进位
    int len = s1.size();
    if(s2.size() > s1.size()) len = s2.size();

    for(i = 0;i < len;i++){
        c[i] = c[i] + a[i] + b[i];
        if(c[i] >= 10){
            c[i+1] = c[i+1] + c[i] / 10;
            c[i] = c[i] % 10;
        }
    }
```

```
    //判断是否多出 1 位
    if(c[len] != 0) len++;

    //逆序将 c 数组拼接乘字符串
    for(i = len-1;i>=0;i--){
        //将 c[i]+'0' 这个 ascii 码强制转换为 char 类型
        r = r + (char)(c[i] + '0');
    }

    return r;
}
int main(){
    //z：代表计算结果，xy 代表 z 的前两项
    string x,y,z;
    int i,n;
    cin>>n;
    //A(n)=A(n-1)*2+A(n-2)
    x = "1";
    y = "2";
    if(n == 1){
        cout<<x;
    } else if(n == 2){
        cout<<y;
    } else{
        //从第 3 项开始递推
        for(i = 3;i <= n;i++){
            z = he(he(y,y),x);
            //修改 xy 的值，逐步向后推导
            x = y;
            y = z;
        }

        cout<<z;
    }
}
```

Pell 数列

```
#include<bits/stdc++.h>//2-1369-3    jiangyf70
using namespace std;

string mult(string s, int b)
{
    int a[500], c[500];
    memset(a, 0, sizeof(a));
    memset(c, 0, sizeof(c));
    for(int i = s.size() - 1; i >= 0; i--) a[s.size() - 1 - i] = s[i] - '0';
    int len = s.size();
    int t = 0;
    for(int i = 0; i < len; i++)
    {
        t += a[i] * b;
        c[i]  = t % 10;
        t /= 10;
    }
    if(t) c[len++] = t;
    string s1;
    for(int i = len -1; i >= 0; i--)
    {
        s1 += c[i] + '0';
    }
    return s1;
}
```

```
string add(string s1, string s2)
{
    int a[500], b[500], c[500];
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));
    memset(c, 0, sizeof(c));
    if(s1.size() < s2.size()) swap(s1, s2);
    for(int i = 0; i < s1.size(); i++) a[s1.size() - 1 - i] = s1[i] - '0';
    for(int i = 0; i < s1.size(); i++) b[s2.size() - 1 - i] = s2[i] - '0';
    int len = s1.size();
    int t = 0;
    for(int i = 0; i < len; i++)
    {
        t += a[i] + b[i];
        c[i] = t % 10;
        t /= 10;
    }
    if(t) c[len++] = t;
    string s3;
    for(int i = len -1; i >= 0; i--)    {    s3 += c[i] + '0';    }
    return s3;
}
int main()
{
    int n;
    cin >> n;
    string a = "1", b = "2", c = "5";
    for(int i = 2; i <= n; i++)
    {
        a = b;      b = c;          c = add(mult(b, 2), a);
       // c = add(add(b, b), a); 只用 add 加法函数也是可以的。
    }
    cout << a << endl;
    return 0;
}
```

小 X 与神牛

```
#include<bits/stdc++.h>//3-1532-1  jiangyf70
using namespace std;
int k = 0, a[20], n, b, d;
int geshu(int y)  //计算 1 的个数
{
    int cnt = 0;
    while(y)
    {
        y = y & (y - 1);  //删除最后一个 1
        cnt++;
    }
    return cnt;
}
bool check(int x)
{
    int y;
    for(int i = 0; i <= k; i++)
    {
        y = a[i] ^ x; //异或，不相同的位都是 1
        if(geshu(y) < d) return 0;
    }
    return 1;
}
int main()
{
    cin >> n >> b >> d;
    for(int i = 1; i <(1 << b); i++)
    {
        if(check(i))     {    k++;      a[k] = i;      }
        if(k == n - 1) break;
    }
    for(int i = 0; i < n; i++)   {    cout << a[i] << " ";   }
    return 0;
}
```

```cpp
#include <bits/stdc++.h>//3-1532-2    javacn
using namespace std;
// 存储符合条件的数对应的二进制
//k 表示 r 数组的长度，默认 0 是符合要求的
int r[20][10],k = 1;
int n,b,d;
// 将 x 转换为 b 位的二进制，判断和 r 数组的 k 个数是否都满足有 >=d 位的不同
bool check(int x){
    int t[10] = {0};// 存储 x 转 2 进制的结果
    int l = 0;
    while(x != 0){
        l++;
        t[l] = x % 2;
        x /= 2;
    }

    // 循环 r 数组的 k 个数
    int c;
    for(int i = 1;i <= k;i++){
        c = 0;
        for(int j = 1;j <= b;j++){
            if(t[j] != r[i][j]) c++;
        }

        if(c < d) return false;
    }

    // 到这个位置，说明编号为 x 的数和 r 数组中的数是友好的
    // 将编号为 x 的数对应的二进制存储到 r 数组
    k++;
    for(int i = 1;i <= b;i++){
        r[k][i] = t[i];
    }
    return true;
}
```

```
int main()
{
    cin>>n>>b>>d;
    cout<<0<<" ";//0 是默认的一个解
    int i = 1;
    while(k < n){
        //判断 i 是否友好
        if(check(i)){
            cout<<i<<" ";
        }

        i++;
    }

    return 0;
}
```

本质：

从 0 开始，找出 N 个数，N 个数转成 B 位的 2 进制，
要求任何两个数都有 >=D 位不同的数。

思路：

1. 从 0 开始将每个符合条件的数转换为 B 位的 2 进制存入二维数组
2. 循环每个数 i，判断是否满足条件：
i 转换为 B 位的二进制要求和二维数组中已经存储的所有数都有 >=D 位不同

1、先找到数组范围，0~ 2 的 b 次方

2、先输出范围内这些数的二进制（测试，后续自行删除）

3、测试无误，先存放一个到答案数组 a 中

4、依次将范围内的数的二进制和 a 数组中的每个元素作比较（详见 check 函数）

5、比较晚记录个数，并输出所有可能结果

```
#include <bits/stdc++.h>//3-1532-3    1989444607
using namespace std;
// 数字转二进制
string numTo2(int n, int b) {
	string res = "";
	while (n) {
		res = char(n % 2 + '0') + res;
		n /= 2;
	}
	int len = b - res.size();
	for (int i = 1; i <= len; i++) {
		res ="0" + res;
	}
	return res;
}
// 判断字符串 s, 是否符合要求
bool check(string s, string a[], int b, int d, int len) {
	// 判断字符串是否已在数组中，返回假
	for (int i = 0; i < len; i++) {
		if (a[i] == s) {
			return false;
		}
	}

	// 如果不在数组中
	// 遍历每个字符串，判断 s 是否满足要求（有 d 个字符不相等）.
	/* 条件：依次和 a 数组中的每个数据作比较，比较不相同的字符个数
			如果有一次不相同的字符的个数不满足，则这个字符串不满足条件
	*/
```

```
        for(int i=0;i<len;i++){
                // 判断不相同的字符的个数
                int x = 0;
                for(int j=0;j<b;j++){
                        if(a[i][j]!=s[j]){
                                x++;
                        }
                }
                // 有一次不满足，返回假
                if(x<d){
                        return false;
                }
        }
        // 上述条件都不成立，返回真
        return true;
}
// 字符串转数字
int sToNum(string s){
        int num=0;
        int p=1;
        for(int i=s.size()-1;i>=0;i--){
                num+=(s[i]-'0')*p;
                p*=2;
        }
        return num;
}
```

```
int main() {
	string a[20], cmp,s2;
	int n, b, d, num = 1, count = 1, p = 1, i, j, k, now;
//	cout<<numTo2(4,5);
	cin >> n >> b >> d;
	for (i = 1; i <= b; i++) {
		num *= 2;
	}
	a[0] = numTo2(0, b);
	for (i = 0; i < num; i++) {
		// 每个 i 二进制和数组里面的数作比较
		// 计算字符串数组中元素的个数
		int len = sizeof(a) / sizeof(a[0]);
		// 需要判断的字符串
		s2 = numTo2(i, b);
		// 如果符合要求，把字符串放入数组中
		if(check(s2, a, b, d, len)){
			a[p]=s2;
			p++;
			count++;
		}
		// 当个数满足要求，退出比较
		if(count==n){
			break;
		}
	}
	// 输出数组中满足条件的值
	for (i = 0; i < p; i++) {
		cout << sToNum(a[i]) << " ";
	}
	return 0;
}
```

蜜蜂路线

```
#include<bits/stdc++.h>//4-1368     javacn
using namespace std;
// 当 n-m 的值较大时需要高精度运算
string he(string s1,string s2){
	string r;// 存放总和
	int a[1000] = {0};
	int b[1000] = {0};
	int c[1000] = {0};

	// 逆序存入整数数组
	int i;
	for(i = 0;i < s1.size();i++){
		a[i] = s1[s1.size() - i - 1] - '0';
	}

	for(i = 0;i < s2.size();i++){
		b[i] = s2[s2.size() - i - 1] - '0';
	}

	// 逐位相加，逐位进位
	int len = s1.size();
	if(s2.size() > s1.size()) len = s2.size();

	for(i = 0;i < len;i++){
		c[i] = c[i] + a[i] + b[i];
		if(c[i] >= 10){
			c[i+1] = c[i+1] + c[i] / 10;
			c[i] = c[i] % 10;
		}
	}
	// 判断是否多出 1 位
	if(c[len] != 0) len++;

	// 逆序将 c 数组拼接乘字符串
```

```
        // 逆序将 c 数组拼接乘字符串
        for(i = len-1;i>=0;i--){
                // 将 c[i]+'0' 这个 ascii 码强制转换为 char 类型
                r = r + (char)(c[i] +  '0' );
        }

        return r;
}

int main(){
    int m,n,i,j;
    cin>>m>>n;
    i = n - m;
    string x,y,z;
    x = "1";
    y = "1";
    if(i==1){
      cout<<x;
      }else if(i==2){
              cout<<y;
      }else{
              for(j=3;j<=i+1;j++){
                      z = he(x,y);
                      x = y;
                      y = z;
              }
              cout<<z;
      }
    return 0;
}
```