高精度运算

高精度运算基础

1268 - 高精度加法

计算 a+b 的值，a,b 皆为不超过 240 位的非负整数。

输入

两个正整数，每行一个。

输出

一个数，代表两个整数的和。

输入	输出
1 2	3

输入	输出
0 0	0

输入	输出
11111111111111111111 22222222222222222222	33333333333333333333

说明

a,b 两个整数都不含前导 0，即：如果 a=100 不可能输入 0100 这样的含有前导 0 的整数。

需要注意的是 a,b 可能为 0。

```cpp
#include <bits/stdc++.h>//1-1268-1 宋老师视频教程
using namespace std;
/*
第一步：用 string 读入高精度整数
第二步：将两个高精度整数逆序存入 a,b 两个整数数组
第三步：从左向右，逐位求和，结果存入 c 数组
第四步：逆序输出结果
*/
string s1, s2; // 高精度整数
int a[250], b[250], c[500];
int i, j, len;
int main() {
	cin >> s1 >> s2;
	// 第二步：将两个高精度整数逆序存入 a,b 两个整数数组
	for(i = 0; i < s1.size(); i++) {
		a[s1.size() - i - 1] = s1[i] - '0';
	}
	for (i = 0; i < s2.size(); i++) {
		b[s2.size() - i - 1] = s2[i] - '0';
	}

	// 第三步：从左向右，逐位求和，结果存入 c 数组 从左向右，逐位进位
	 // 加法的次数，取决于两个整数的较长的字符串
	len = s1.size();
	if (s2.size() > s1.size()) {
		len = s2.size();
	}

	for (i = 0; i < len; i++) {	// 逐位相加
		c[i] = a[i] + b[i];
	}

	// 逐位进位
```

```
    // 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;
        }
    }

    // 第四步：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++;
    }

    // 逆序输出结果
    for(i = len - 1; i >= 0; i--) {
        cout << c[i];
    }

    return 0;
}
```

```cpp
#include <bits/stdc++.h>//1-1268-2    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;
}
```

```
/* 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>//1-1268-3    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

*/

高精度加法

```
#include<iostream>//1-1268-4    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;
}
```

1269 - 高精度减法

高精度减法，求 a-b。a,b 都是不超过 240 位的非负整数。

输入

两个非负整数，每行一个。

输出

一个整数，代表两个整数相减之后的结果。

输入

333333333333333333333333333333333333333333

222222222222222222222222222222222222222222

输出

1111111111111111111111111111111111111111111

```
#include <bits/stdc++.h>//2-1269-1  宋老师视频教程
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;
    for(i = 0;i < s1.size();i++){ // 将 s1 和 s2 逆序存入整数数组
        //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';
    }

    len = s1.size();  // 逐位相减

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

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

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

高精度减法

```
#include<iostream>//2-1269-2  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;
}
```

1286 - 高精度乘单精度

高精度乘单精度，a × b。 a 是一个很大的非负整数，但不超过 240 位，b 是一个非负整数不超过 10000 ，求 a × b 。

输入

两行数字。

第一行是 a ，第二行是 b 。

输出

一行，输出 a × b 的计算结果。

输入

111111111111111111111111111111111111

10

输出

1111111111111111111111111111111111110

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[250],b,c[250];
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;
}
```

1287 - 高精度乘

高精度乘，求两个很大的非负整数相乘的结果。

输入

2 个非负整数，每个一行，每个整数不超过 240 位。

输出

一个整数，表示相乘的结果。

输入

111111111111111111111111

222222222222222222222222

输出

246913580246913580246913530864197530864197530864 2

```
#include <bits/stdc++.h>//4-1287-1 宋老师视频编程
using namespace std;
string s1, s2;
int a[250], b[250], c[500];
int i, j, p;// p=0，防止全 0 情况
int main() {
	cin >> s1 >> s2;	// 逆序将 s1 和 s2 存入 a,b 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];

                        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，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;
}
```

```cpp
#include <bits/stdc++.h>//4-1287 -2 千问
using namespace std;
string s1, s2;
int a[250], b[250], c[500];
int i, j, p;
int main() {
	cin >> s1 >> s2;
	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';
	}

	for(i = 0; i < s1.size(); i++) {	// 逐位相乘并累加
		for(j = 0; j < s2.size(); j++) {
			c[i + j] += a[i] * b[j];
		}
	}
	for(i = 0; i < s1.size() + s2.size(); i++) {	// 统一处理进位
		if(c[i] >= 10) {
			c[i + 1] += c[i] / 10;
			c[i] %= 10;
		}
	}
	p = 0;	// 寻找最高非零位
	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];	} // 输出结果
	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>//4-1287-3  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;
}
```

1271 - 高精度整数除法

求 a/b 的结果。

已知 a，b 为 10^8 范围内的非负整数，求 a/b 保留前 n 位小数商的结果。

输入

读入三个整数 a b n。

输出

输出一行数字。

输入

97 61 50

输出

1.59016393442622950819672131147540983606557377049180

```
#include <bits/stdc++.h>//5-1271-1 宋老师视频教程
using namespace std;

int main() {
	long long a, b, i, n, t;
	cin >> a >> b >> n;

	cout << a / b << ".";
	t = a % b;

	for (i = 1; i <= n; i++)
	{
		t = t * 10;
		cout << t / b;
		t = t % b;
	}

	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
	t = a % b;  // 先求得余数 36
	for(i=1;i<=n;i++){
		t = t * 10;		// 余数 *10
		cout<<t/b;		// 小数部分
		t = t % b;
	}
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//5-1271-3 jiangyf70 高精度整数除法
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;
}
```

1280 - 求 2 的 n 次方

求 2 的 n 次方。（0 ≤ n ≤ 100）

输入

从键盘读入一个整数 n;

输出

请输出 2 的 n 次方;

输入

100

输出

1267650600228229401496703205376

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

		for (j = 0; j < k; j++)   // 逐位进位
		{
			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 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>//6-1280-2     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;
}
```

1281 - 求 2+2*2+2*2*2+…+2*2*2*….*2

求 2+2*2+2*2*2+…+2*2*2*…*2 的和是多少？最后一项有多少 2 相乘由键盘读入的 n 决定（1 ≤ n ≤ 100）。

比如：n=3，那么 s=2+2*2+2*2*2=14。

输入

从键盘读入一个整数 n（1 ≤ n ≤ 100）。

输出

输出求出的和。

输入

3

输出

14

高精度 * 整数 以及 高精度求和的结合。

```
#include <bits/stdc++.h>//7-1281-1    宋老师视频教程
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
            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;
}
```

求 2+2*2+2*2*2+…+2*2*2*….*2

```
#include<bits/stdc++.h>//7-1281-2    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>//7-1281-3    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>//7-1281-4    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;
}
```

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;
}
```

```cpp
#include <bits/stdc++.h>//3-1532-1    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

小 X 与神牛

```cpp
#include<bits/stdc++.h>//3-1532-2  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;
}
```

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;
}
```