1. 只出现一次的数

给定n个数字，其中有且仅有一个数字出现一次，其他数字都会出现两次。请找出那个只出现一次的数字。要求使用位运算进行解答，且不使用额外的空间。

```
int n, m, res = 0;
cin >> n;
for(int j=0 ; j<n; j++){
    cin >> m;
    res ^= m;
}
cout << res;
```

2.电话号码加密

某城市的电话号码有八位，现有一套加密算法，加密过程分为两步：

第一步：对于每位数字，首先计算5次方后再加5，然后用结果除以10的余数代替该数字，如6用1代替（(6*6*6*6*6+5)%10=1）；

第二步：将所有的0用9替换，并将所有数字循环右移3位，如76543210变成21976543。

请你设计C/C++程序根据原电话号码给出加密后的号码。

（⚠禁止使用数组，禁止使用STL库中提供的函数）

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main () {
    int a[8];
    for(int j=0; j<8; j++){
        int ch = getchar() - '0';
        ch = ((int)pow(ch, 5) + 5) % 10;
        if (ch == 0) ch = 9;
        a[(j + 3) % 8] = ch;
    }
    for(int j=0; j<8; j++) cout << a[j];
    return 0;
}
```

3. 排名

有 n 位脱口秀演员参加了一个脱口秀年度演出，每位演员的表演时间不同。现按首场演出的出场顺序，将每位演员的表演时间存储在一个数组中。请你设计C/C++程序，按首场演出的出场顺序，给出这些演员表演时间的名次。

输入格式（2行）

第一行，一个整数 n（1＜n＜100，代表演员数）

第二行，n 个正整数（代表表演时间）

输出格式

一行，n 个整数（代表名次）

思路一：先排序、再取位次

先进行冒泡排序，从大到小

```
void MySort(int a[], int s, int e)
{
	for (int i = 0; i < e - 1 - s; ++i){
		for (int j = s; j < e - 1 - i; ++j){
			if (a[j] < a[j + 1]){
				int tmp = a[j];
				a[j] = a[j + 1];
				a[j + 1] = tmp;
			}
		}
	}
}
```

思路一：全部排序、再取位次

```
int main()
{
    int n, a[100], b[100];
    cin >> n;
    for (int i = 0; i < n; ++i){
        cin >> a[i];
        b[i] = a[i];
    }
    MySort(b, 0, n);
    for (int i = 0; i < n; ++i){
        for (int j = 0; j < n; ++j){
            if (a[i] == b[j]){
                cout << j + 1 << " ";
                break;
            }
        }
    }
    return 0;
}
```

思路二：扫描计数、确定位次

```
#define N 100
int main() {
      int a[N], n;
      cin >> n;
      for (int i = 0; i < n; ++i) cin >> a[i];
      int b[N];
      for (int i = 0; i < n; ++i) {
            int pos = 1;
            for (int j = 0; j < n; ++j) {
                  if (a[j] > a[i]) ++pos;
            } //有几个数超过 a[i]
            b[i] = pos; // a[i]对应的序号
      }
      for (int i = 0; i < n; ++i) cout << b[i] << ' ';
      return 0;
}
```

思路二：扫描计数、确定位次

```
#define N 100
int main() {
	int a[N], n;
	cin >> n;
	for (int i = 0; i < n; ++i) cin >> a[i];
	for (int i = 0; i < n; ++i) {
		int pos = 1;
		for (int j = 0; j < n; ++j) {
			if (a[j] > a[i]) ++pos;
		}
		cout << pos << ' ';
	}
	return 0;
}
```

4.多数元素

请你设计C/C++程序，用一个数组存储 n 个整数(int)，并找到其中的多数元素。多数元素是指在数组中出现次数大于[n/2]（向下取整）的元素。给定的数据总是存在多数元素。

输入格式（2行）

第一行，一个整数 n（1＜n＜100，代表整数的个数）

第二行，n 个整数（代表元素）

输出格式

一行，一个整数（代表那个多数元素）

思路一：全部排序、取[n/2]值

1 2 1 3 1 1 2

1 1 1 1 2 2 3

```
cout << arr[n / 2];
```

```
int main()
{
    int n, a[100];
    cin >> n;
    for (int i = 0; i < n; ++i){
        cin >> a[i];
    }
    MySort(a, 0, n);
    cout << a[n/2] << endl;
    return 0;
}
```

思路二：扫描计数、判断出现次数是否>[n/2]

```
int main()
{
    int n, a[100];
    cin >> n;
    for (int i = 0; i < n; ++i) cin >> a[i];
    for (int i = 0; i < n; ++i){
        int count = 0;
        for (int j = 0; j < n; ++j)
            if (a[i] == a[j])
                ++count;
        if (count > n / 2){
            cout << a[i];
            break;
        }
    }
    return 0;
}
```

思路三：“元素消消乐”

因为该元素出现次数超过数组长度一半，
所以该数字的个数 - 其他数字的个数总和 >=

1。

```
int majorityElement(int a[], int n)
{
      int cnt = 0, temp = 0;
      for(int i=0; i < n; ++i) {
            if(cnt == 0)
                  temp = a[i];    //重置 目标元素
            if(a[i] == temp)
                  ++cnt;
            else                  //出现其他元素
                  --cnt;          //抵消一次该元素
      }
      return temp;
}
```

思路三：“元素消消乐”（不用数组）

因为该元素出现次数超过数组长度一半，
所以该数字的个数 - 其他数字的个数总和 >=

1。

```
int n;
cin >> n;
int d, cnt = 0,temp = 0;
for(int i=0; i < n; ++i) {
      cin >> d;
      if(cnt == 0)
            temp = d;
      if(d == temp)
            ++cnt;
      else
            --cnt;
}
cout << temp;
```

5. [选做]第 k 大的数

请你设计C/C++程序，求解一个整数（int）**集合**中的第 k 大的数。

输入格式（2行）

第一行，一个整数 n（1＜n＜100，代表整数集合大小）

第二行，n 个整数（代表集合中的元素）

第三行，一个整数 k（1＜k≤n）

输出格式

一行，一个整数（代表第 k 大的数）

思路一：全部排序、取[k-1]值

```
#define N 100
int a[N];

int n, k;
cin >> n >> k;
for(int i=0; i < n; ++i)
      cin >> a[i];

cout << kthElement(a, n, k);
```

```
int kthElement(int a[], int n, int k){
      MySort(a, 0, n); //冒泡排序
      return a[k-1];
}
```

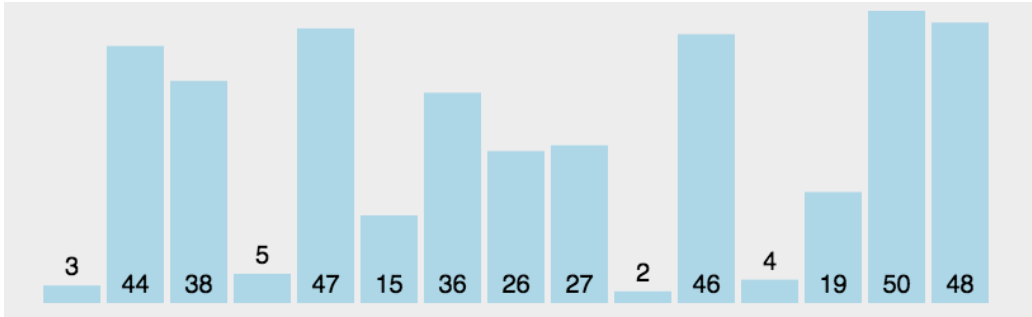
思路一：全部排序、取[k-1]值

```
#define N 100
int a[N];

int n, k;
cin >> n >> k;
for(int i=0; i < n; ++i)
      cin >> a[i];

cout << kthElement(a, n, k);
```

```
int kthElement(int a[], int n, int k){
      MySort(a, 0, n); //冒泡排序
      return a[k-1];
}
```

思路一：全部排序、取[k-1]值

```
int kthElement(int a[], int n, int k){
      for(int i = 0; i < n-1; ++i){
            int min = i;
            for(int j = i+1; j < n; ++j)
                  if(a[min] >= a[j])
                        min = j;
            if(min != i){
                  int temp = a[min];
                  a[min] = a[i];
                  a[i] = temp;
            }
      }
      return a[n-k];
}//从小到大排序，选择排序
```



思路一：全部排序、取[k-1]值

```
int kthElement(int a[], int n, int k){
      for(int i = n; i > 1; --i){
            int min = 0;
            for(int j = 1; j < i; ++j)
                  if(a[min] >= a[j])
                        min = j;
            if(min != i-1){
                  int temp = a[min];
                  a[min] = a[i-1];
                  a[i-1] = temp;
            }
      }
      return a[k-1];
} //从大到小排序，选择排序
```

思路二：部分排序

```
for(int j = 0; j < k; ++j)
{
      int max = 0;
      for(int i = 0; i < n-1-j; ++i)
      {     if(a[i+1] >= a[max])
                  max = i+1;
      }
      if(max != n-1-j)
      {
            int temp = a[n-1-j];
            a[n-1-j] = a[max];
            a[max] = temp;
      }
}
return a[n-k];
```

提前结束排序

思考：利用快排思想怎么做？

思路二：部分排序

```
int quickSort(int* num, int left, int right){
    int i = left;
    int j = right;
    int base = num[left];
    while(i < j){
        while(i < j && num[j] >= base) j--;
        if(i < j) num[i++] = num[j];
        while(i < j && num[i] < base) i++;
        if(i < j) num[j--] = num[i];
    }
    num[i] = base;
    quick_sort(num, left, i-1);
    quick_sort(num, i+1, right);
    return i;
}
```

锚点位置和k进行比较

思路二：部分排序

```
int kthElement(int a[], int start, int end, int k){
    int i = quickSort(a, start, end);
    if (i == k) return a[k];
    if (i < k){
        return kthElement(a, i + 1, end, k);
    }
    return kthElement(a, start, i - 1, k);
}

int main(){
    int n, k, a[100]; cin >> n;
    for (int i = 0; i < n; ++i) cin >> a[i];
    cin >> k;
    cout << kthElement(a, 0, n-1, n-k) << endl;
    return 0;
}
```

思路三：扫描计数、判断位次是不是k

```
for(int j=0; j < n; ++j)
{
      int count = 0;
      for(int m=0; m < n; ++m)
      {
            if(a[j] >= a[m])
                  ++count;
      } //a[j]能超过几个数
      if(count == n-k+1)
            return a[j];
}
```

前提是存在第 k 大的数据，比如 2 2 2 2 1当中就不存在第三大的数据

不排序