选择题部分：

进制：进制转换

十进制转二进制：整数部分除二取余法，小数部分乘二取整

1.

```
int n;
string s;
cin>>n;
while(n!=0){
  s=char(n%2+48)+s;
  n/=2;
}
cout<<s;
```

2.

```
void f(int n){
  if(n==0) return ;
  f(n/2);
  cout<<n%2;
}
```

二进制转十进制：按位权求和

```
string s;
int sum=0;
cin>>s;
for(int i=0;i<s.length();i++){
  sum=sum*2+(s[i]-48); // sum=sum+(s[i]-48)*pow(2,s.length()-i-1);
}
cout<<s;
```

快速转化法：四位二进制=一位十六进制

二进制：位运算按照优先级 ~ 取反，<< 左移 >>右移 ，& 按位与 ,^ 按位异或 ，| 按位或

存储

单位转换：8bit = 1B ，1024B=1KB ，MB ，GB ,TB
图像存储：真彩色位图是 24 位，分辨率*位色=单位是位

二进制三码：原码，补码，反码

原码除符号 0 变 1,1 变 0 = 反码
反码+1 = 补码
计算机存储是以二进制补码存储，三码只针对负数

数据结构

向量：g.push_back() 末尾添加元素 ，g.size() 元素个数

栈：q.push() 栈顶入栈 ， q.pop() 栈顶出栈 ， q.size() 栈元素个数 ， q.top() 栈顶 ， q.empty() 判断栈是否为空，空返回 true ， 先进后出

队列：q.push() 队尾入队 ， q.pop() 对首出队 ， q.size() 队元素个数 ， q.front() 队首， q.back() 队尾 ， q.empty() 判断队是否为空，空返回 true ， 先进先出

链表：单向链表和双向链表,单向链表指只能从前往后访问。

在节点 a 后面插入 b

```
b.l=a;
b.r=a.r;
a.r.l=b;
a.r=b;
```

删除节点 a 后面的元素

```
a.r.r.l=a
a.r=a.r.r
```

算法

十大排序:复杂度(全部)，稳定性(全部)，代码(冒泡，选择，插入，快速，归并，计数)
复杂度：

排序方法	时间复杂度（平均）	时间复杂度（最坏）	时间复杂度（最好）	空间复杂度	稳定性
插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
希尔排序	$O(n^{1.3})$	$O(n^2)$	$O(n)$	$O(1)$	不稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(1)$	不稳定
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
快速排序	$O(nlog_2n)$	$O(n^2)$	$O(nlog_2n)$	$O(nlog_2n)$	不稳定
归并排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(n)$	稳定
计数排序	$O(n+k)$	$O(n+k)$	$O(n+k)$	$O(n+k)$	稳定
桶排序	$O(n+k)$	$O(n^2)$	$O(n)$	$O(n+k)$	稳定
基数排序	$O(n*k)$	$O(n*k)$	$O(n*k)$	$O(n+k)$	稳定

```cpp
//冒泡排序：复杂度 O(n^2),稳定排序
#include<iostream>
using namespace std;
int main(){
  int n,a[10005];
  cin>>n;
  for(int i=1;i<=n;i++)
    cin>>a[i];

  for(int i=n;i>=2;i--){ //执行轮数只有 n-1 论
    for(int j=1;j<=i-1;j++){
      if(a[j]>a[j+1]) //比较相邻的两个数
        swap(a[j],a[j+1]);//交换，交换 a[j]和 a[j+1]的值

    }
  }

  for(int i=1;i<=n;i++)
    cout<<a[i]<<' ';
  return 0;
}

//选择排序：复杂度 O(n^2),不稳定排序
#include<iostream>
using namespace std;
//全局变量
int main(){
  int n,a[10005];
  cin>>n;
  for(int i=1;i<=n;i++)
    cin>>a[i];

  for(int i=1;i<=n-1;i++){
    int min=0x3f3f3f3f,mint;//mint 记录最小值的下标
    //记录最小值的是 min
    for(int j=i;j<=n;j++){
      if(a[j]<min){
        min=a[j];
        mint=j;//记录最小值下标
      }
    }
    swap(a[i],a[mint]);
  }

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

```cpp
//插入排序：复杂度 O(n^2),稳定排序
//--------三大基本排序
#include<iostream>
using namespace std;
//全局变量
int main(){
  int n,a[10005];
  cin>>n;
  for(int i=1;i<=n;i++)
    cin>>a[i];

  for(int i=2;i<=n;i++){
    int j=i;
    while(a[j-1]>a[j] && j>=1){
      swap(a[j-1],a[j]); //交换函数
      j--;
    }
  }

  for(int i=1;i<=n;i++)
    cout<<a[i]<<' ';
  return 0;
}

//快速排序
#include<iostream>
using namespace std;

const int N = 1e4 + 5, M = 1e4 + 5;
int a[N];
int n;

void show()
{
  for(int i = 1; i <= n; i++)
    cout << a[i] << ' ';
}

void quickSort(int l, int r)
{
  if(l >= r)
    return;
  int t = a[l]; //  基准值
  int i = l, j = r; //  哨兵
  while(i < j){
    while(a[j] > t)
      j--;
    while(a[i] <= t && i < j)
      i++;
```

```
    if(i != j)
      swap(a[i], a[j]);
  }
  swap(a[l], a[i]);

  quickSort(l, i - 1);
  quickSort(i + 1, r);
}

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

  return 0;
}

//归并排序: 复杂度 O(n*logn) 2*10^5 , 稳定排序
#include<iostream>
using namespace std;

int n,a[10005],tmp[10005];//tmp 是合并后的序列
void sort(int l,int r){ //l,r 要排序的区间
  if(l>=r) return ;
  int mid=(l+r)/2;//取中间值
  sort(l,mid);//左边
  sort(mid+1,r);//右边
  //回来开始合并，合并 2 个有序序列

  l=1,r=8  4 5 7 8,1 2 3 6
  int i=l,j=mid+1,k=1;
  //i 访问前半段序列，j 访问后半段序列，k 访问 tmp 数组
  while(i<=mid && j<=r){ //合并
    if(a[i]<a[j]){ //比较开头
      tmp[k]=a[i];
      k++;i++;
    }else{
      tmp[k]=a[j];
      k++;j++;
    }
  }
  tmp[1]=4,tmp[2]=5;tmp[3]=7;

  //按照顺序放入
  while(i<=mid){ //前半段序列有多
    tmp[k]=a[i];
```

```
    k++;i++;
  }
  tmp[4]=8;
  while(j<=r){ //后半段序列有多
    tmp[k]=a[j];
    k++;j++;
  }
  //tmp 就是合并过后有序的序列
  for(int i=l;i<=r;i++)
    a[i]=tmp[i-l+1];
  //我们排序是 a 数组，把排序号的 tmp 赋值回去
  a[1]=4,a[2]=5,a[3]=7,a[4]=8;
  return ;
}

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

  sort(1,n);

  /*
  for(int gap=n/2;gap>=1;gap/=2){//控制增量
    for(int i=gap;i<=n;i++){ //如果 gap=1，带入就是插入排序
      int j=i;
      while(a[j]<a[j-gap] && j>=gap){
        swap(a[j],a[j-gap]);
        j-=gap;
      }
    }
  }*/

  for(int i=1;i<=n;i++)
    cout<<a[i]<<' ';
  return 0;
}

//计数排序
#include<iostream>
using namespace std;
int n,a[10005],to[10005];//桶数组，用于计数
int main(){
  cin>>n;
  for(int i=1;i<=n;i++){
    cin>>a[i];
    to[a[i]]++;
    //最先进入 to 数组肯定是最靠前的
  }
```

```
  //O(n+k),稳定排序
  for(int i=1;i<=10000;i++) //取决于数字的大小， k=max(a[i])
    //默认最先输出的就是最先进入的
      while(to[i]!=0){ //n
        cout<<i<<' ';
        to[i]--;
      }
   return 0;
}
```

数学

组合数学：排列，组合，容斥，抽屉，概率

排列：选择的数存在位置关系， A(n,m)=A(5,3)=5*4*3;
组合：选择的数不存在位置关系， C(n,m)=C(5,3)=5*4*3/1/2/3;
容斥：A∪B=A+B-A∩B ， A+B+C-A∩B-A∩C-B∩C+A∩B∩C=A∪B∪C ,∪ 取全部，∩取公共部分
抽屉：如果 n 大于 m，则至少有一个抽屉存在 n/m+1
概率：方案数/总数=概率

数论：最大公约数(代码)，质数筛法(埃式筛法，欧拉筛法)

```
//最大公约数
#include<iostream>
using namespace std;
int gcd(int a,int b)
{
  if(a%b==0) return b;
  return gcd(b,a%b);
}
int main()
{
  int a,b;
  cin>>a>>b;
  cout<<gcd(a,b);
  return 0;
}
```

```
//质数筛法
  //埃氏筛法：每个数筛自己的倍数
#include <iostream>
#include <vector>
#include <cstdio>
using namespace std;

const int N = 1e7+5;
int n;
bool isPrime[N];
vector<int> primes;
```

```
int main() {
  cin>>n;
  for(int i=1; i<=n; i++)//默认都是质数
    isPrime[i] = true;


  for(int i=2; i<=n; i++) {
    if (isPrime[i]) {//不是合数，也就是质数
      primes.push_back(i);
      for(int j=2*i; j<=n ; j=j+i)
          isPrime[j] = false;
    }
  }

  cout<<primes.size()<<endl;
  for(int i=0; i<primes.size(); i++)
    printf("%d ", primes[i]);
  cout<<endl;
  return 0;
}

  //欧拉筛法：每个数 x 筛去 x * <=最小质因子的质数
#include <iostream>
#include <vector>
#include <cstdio>
using namespace std;

const int N = 1e7+5;
int n;
bool isPrime[N];
vector<int> primes;

int main() {
  cin>>n;
  for(int i=1; i<=n; i++)//默认都是质数
    isPrime[i] = true;


  for(int i=2; i<=n; i++) {
    if(isPrime[i])
      primes.push_back(i);
    for(int j=0; j<primes.size(); j++) {
      int p = primes[j];
      if(i*p > n) break;
      isPrime[i * p] = false;
      if(i%p == 0) break;
    }
  }
```

```
  cout<<primes.size()<<endl;
  for(int i=0; i<primes.size(); i++)
    printf("%d ", primes[i]);
  cout<<endl;
  return 0;
}
```

普及组算法

C 语言的代码：
#include<cstdio> c 语音标准头文件 scanf() 输入 ， printf() 输出
#include<iostream> max(2,3)=2 ， min(2,3)=2 ， swap(a[i],a[i+1]) 交换
#include<cmath> sqrt(9)=3 开根，pow(2,3)=8 求幂,abs(-3)=3 绝对值,
#include<iomanip> fixed<<setprecision(2) 保留小数点后 2 位，有四舍五入
#include<cstring> s.length() 字符串长度 strlen(s) 字符数组长度 ，
#include<algorithm> sort()排序
#include<bits/stdc++.h> 万能头，包含所有头文件,只包含 c++和 c

scanf("%d",&n); //输入一个整数 n， %d 整数，%lld 长整数， %f float 小数 ，%llf double ,%c 字符， %s 字符串
快速读入
&n ， &地址 ， 输入字符串和数组是不许带 &

  n=3
printf("%");
快速输出
printf("%d ",n)

  gets(s)//输入字符串
  puts(s)//输出字符串

 getchar();//输入字符，接受回车

  endl,换行符也是字符

二分,n 选 m 排列，n 选 m 组合，迷宫搜索，背包，深搜，广搜，三序遍历，树的父亲表示法，树的孩子表示法，邻接表，邻接矩阵，拓扑序列，环 ，