变量 a 是一个 64 位有符号的整数，初始值用 16 进制表示为：0x7FFFFFFFFFFFFFFF;变量 b 是一个 64 位有符号的整数，初始值用 16 进制表示为：0x8000000000000000。则 a+b 的结果用 10 进制表示为多少？

正确答案:B 你的答案:空(错误)

1
-1
2^63+2^62+…+2^2+2^1+2^0
-(2^63+2^62+…+2^2+2^1+2^0)

TCP 建立连接的三次握手中，第二次握手发送的包会包含的标记，最正确的描述是？

正确答案:B 你的答案:空(错误)

ACK
SYN,ACK
SYN,PSH
SYN

栈是先进后出的数据结构。给定一个大小为 3 的初始状态为空的栈，已知一组数据经过这个栈后，最终的数据顺序依次为：1324，问原始的进栈数据不可能是以下的那组?

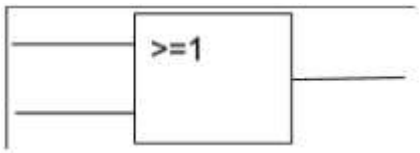
正确答案:C 你的答案:空(错误)

2314
1423
4231
3124

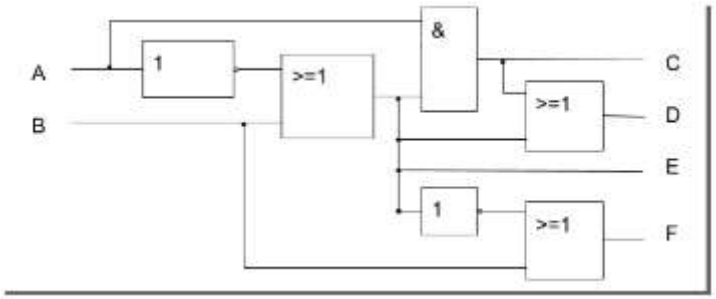
电路中其中三个门电路非门，与门，或门的示意图及性质分别如下所示：
非门，使输入的电平编程相反电平：

与门，使输入两个高电平，输出高电平，其他情况下输出低电平：

或门。当且仅当输入两个低电平时，输出低电平，否则输出高电平：



现在对以下的电路中的 A 和 B 引脚分别持续输入一个高电平（1）和一个低电平（0），问最终电路的引脚 C、D、E、F 分别输出的电平是什么？



正确答案:D 你的答案:空(错误)

C=0,D=1,E=0,F=1
C=1,D=1,E=1,F=0
C=1,D=1,E=0,F=1
C=0,D=0,E=0,F=1

下面这个 C++进程运行时，描述正确的是（）

```
int a = 1;
const char* str = "abc";

class Obj {};

int f(int lhs, int rhs) {
    Obj* obj = new Obj();
    Obj obj2;
    int aa = a;
    delete obj;
    return aa + lhs + rhs;
}

int main() {
    int* b = new int(2);
    int c = f(a, *b);
    delete b;
}
```

正确答案:A 你的答案:空(错误)

保存在堆中的数据有：*b,*obj;保存在栈中的数据有：c,lhs,rhs,obj2,aa
保存在堆中的数据有：*b,*obj;保存在栈中的数据有：str,c,lhs,rhs,obj2,aa
保存在堆中的数据有：*b,*obj2;保存在栈中的数据有：c,lhs,rhs,obj,aa
保存在堆中的数据有：*b,*obj,*str;保存在栈中的数据有：c,lhs,rhs,aa

操作系统中可以使用 LRU(leastrecentlyused)内存淘汰旧数据的策略，如果内存需要加载新数据但空间又不足，则会按照最近访问时间进行排序，并将最老的数据淘汰，假设现在内存空间大小为 5，原本内存中没有数据，对内存中数据的访问顺序如下：
1,2,5,3,4,6,1,4,3,6,7,8,3,9
正确答案:C 你的答案:空(错误)

缺页次数：9
缺页次数：4
缺页次数：10
缺页次数：5

下面的程序中，int32-t 表示一个有符号的 32 位整数，程序的入口是 main 函数，问最终 res 的结果是多少？

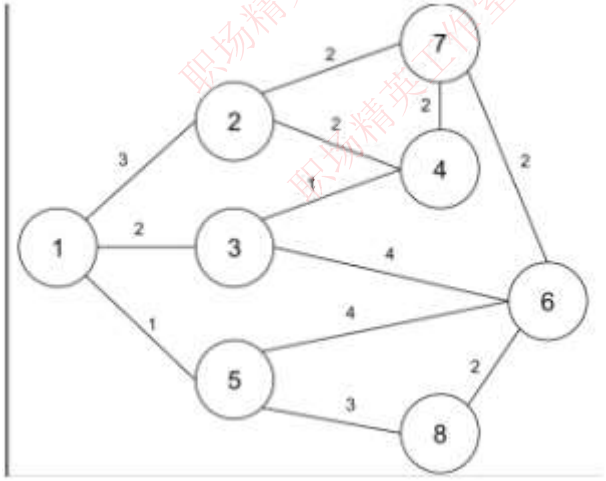
```
int32_t f(int32_t a, int32_t b) {
    while (a + b > 0) {
        a = a + 1;
        b = b - 1;
    }
    return a + b;
}

int32_t main() {
    int32_t res = f(1, 0);
}
```

正确答案:D 你的答案:空(错误)

-(2^31+2^30+…+2^2+2^1+2^0)
0
-1
程序会死循环

给定一个如下所示的图，图中的边代表了两个节点间的距离。如果使用迪杰斯特拉算法对节点 1 和节点 8 求最短路径，则当完成计算时，算得节点 1 到节点 8 的最短路径是？同时当完成节点 1 到节点 8 的最短路径计算时，节点 1 到哪些节点（除了 1 和 8）的最短路径也已经计算完毕？（ ）



正确答案:B 你的答案:空(错误)

最短路径： 7 ； 已经算得最短路的节点： 3 ， 5 ， 6

最短路径：４；已经算得最短路的节点：５
最短路径：４；已经算得最短路的节点：２，３，５，４
最短路径：４；已经算得最短路的节点：５，６

x86CPU 在实模式下解释代码时看到一个地址为 2330H:5041H,请问它最终在内存中要找的地址是多少？

正确答案:A 你的答案:空(错误)

28341H
5374H
52740H
7371H

有三个程序Ｊ１，Ｊ２，Ｊ３。程序在单核ＣＰＵ执行时，三个程序需要的资源如下所示：

程序	CPU 占用时间（ms）	IO 占用时间（ms）	执行顺序	优先级
J1	40	60	先 CPU 计算后 IO	高
J2	20	50	先 IO 后 CPU 计算	中
J3	30	20	先 CPU 计算后 IO	低

优先级高的程序可以抢占优先级低的程序的ＣＰＵ，但不能抢占ＩＯ。问当所有任务执行完毕时，共消耗的时间是？

正确答案:D 你的答案:空(错误)

170m s
160m s
120m s
130m s

头条的 2017 校招开始了！为了这次校招，我们组织了一个规模宏大的出题团队，每个出题人都出了一些有趣的题目，而我们现在想把这些题目组合成若干场考试出来，在选题之前，我们对题目进行了盲审，并定出了每道题的难度系统。一场考试包含 3 道开放性题目，假设他们的难度从小到大分别为 a,b,c，我们希望这 3 道题能满足下列条件：
a<=b<=c
b-a<=10
c-b<=10
所有出题人一共出了 n 道开放性题目。现在我们想把这 n 道题分布到若干场考试中（1 场或多场，每道题都必须使用且只能用一次），然而由于上述条件的限制，可能有一些考试没法凑够 3 道题，因此出题人就需要多出一些适当难度的题目来让每场考试都达到要求，然而我们出题已经出得很累了，你能计算出我们最少还需要再出几道题吗？

```
#include<iostream>
#include<vector>
#include<algorithm>
```

```
usingnamespacestd;
intmain()
{
intn;
while(cin>>n)
{
vector<int>vec(n,0);
for(auto&i:vec)
cin>>i;
sort(vec.begin(),vec.end());
intcount=0;
intj=1;
for(inti=1;i<vec.size();i++)
{
if(j==3)
{
j=1;
continue;
}
if(vec[i]-vec[i-1]<=10)
j++;
elseif(vec[i]-vec[i-1]<=20&&j==1)
{
count++;
i++;
}
elseif(j==1)
count=count+2;
elseif(j==2)
{
j=1;
count++;
}
}
count+=3-j;
cout<<count<<endl;
}
return0;
}
```

给定整数 m 以及 n 各数字 A1,A2,..An，将数列 A 中所有元素两两异或，共能得到 n(n-1)/2 个结果，请求出这些结果中大于 m 的有多少个。

```
import java.util.Scanner;
```

```
public class Main {
    private static class TrieTree {
        TrieTree[] next = new TrieTree[2];
        int count = 1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()){
            int n = sc.nextInt();
            int m = sc.nextInt();
            int[] a = new int[n];
            for (int i = 0; i < n; i++) {
                a[i] = sc.nextInt();
            }
            System.out.println(solve(a, m));
        }
    }

    private static long solve(int[] a, int m) {
        TrieTree trieTree = buildTrieTree(a);
        long result = 0;
        for (int i = 0; i < a.length; i++) {
            result += queryTrieTree(trieTree, a[i], m, 31);
        }
        return result / 2;
    }

    private static long queryTrieTree(TrieTree trieTree, int a, int m, int index) {
        if(trieTree == null)
            return 0;

        TrieTree current = trieTree;
        for (int i = index; i >= 0; i--) {
            int aDigit = (a >> i) & 1;
            int mDigit = (m >> i) & 1;
            if(aDigit == 1 && mDigit == 1) {
                if(current.next[0] == null)
                    return 0;
                current = current.next[0];
            } else if (aDigit == 0 && mDigit == 1) {
                if(current.next[1] == null)
                    return 0;
```

```
                current = current.next[1];
            } else if (aDigit == 1 && mDigit == 0) {
                long p = queryTrieTree(current.next[1], a, m, i - 1);
                long q = current.next[0] == null ? 0 : current.next[0].count;
                return p + q;
            } else if (aDigit == 0 && mDigit == 0) {
                long p = queryTrieTree(current.next[0], a, m, i - 1);
                long q = current.next[1] == null ? 0 : current.next[1].count;
                return p + q;
            }
        }
        return 0;
    }

    private static TrieTree buildTrieTree(int[] a) {
        TrieTree trieTree = new TrieTree();
        for (int i = 0; i < a.length; i++) {
            TrieTree current = trieTree;
            for (int j = 31; j >= 0; j--) {
                int digit = (a[i] >> j) & 1;
                if(current.next[digit] == null) {
                    current.next[digit] = new TrieTree();
                } else {
                    current.next[digit].count ++;
                }
                current = current.next[digit];
            }
        }
        return trieTree;
    }
}
```