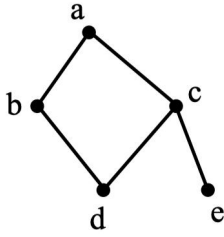


2021 入门组

1. 以下不属于面向对象程序设计语言的是 ()。
A. C++ B. Python C. Java D. C
2. 以下奖项与计算机领域最相关的是 ()。
A. 奥斯卡奖 B. 图灵奖 C. 诺贝尔奖 D. 普利策奖
3. 目前主流的计算机储存数据最终都是转换成 () 数据进行储存。
A. 二进制 B. 十进制 C. 八进制 D. 十六进制
4. 以比较作为基本运算, 在 N 个数中找出最大数, 最坏情况下所需要的最少的比较次数为 ()。
A. N^2 B. N C. $N-1$ D. $N+1$
5. 对于入栈顺序为 a,b,c,d,e 的序列, 下列 () 不是合法的出栈序列。
A. a,b,c,d,e B. e,d,c,b,a C. b,a,c,d,e D. c,d,a,e,b
6. 对于有 n 个顶点、 m 条边的无向连通图 ($m > n$), 需要删掉 () 条边才能使其成为一棵树。
A. $n-1$ B. $m-n$ C. $m-n-1$ D. $m-n+1$
7. 二进制数 101.11 对应的十进制数是 ()。
A. 6.5 B. 5.5 C. 5.75 D. 5.25
8. 如果一棵二叉树只有根结点, 那么这棵二叉树高度为 1。请问高度为 5 的完全二叉树有 () 种不同的形态?
A. 16 B. 15 C. 17 D. 32
9. 表达式 $a*(b+c)*d$ 的后缀表达式为 (), 其中 $*$ 和 $+$ 是运算符。
A. $**a+bcd$ B. $abc+*d*$ C. $abc+d**$ D. $*a*+bcd$
10. 6 个人, 两个人组一队, 总共组成三队, 不区分队伍的编号。不同的组队情况有 () 种。
A. 10 B. 15 C. 30 D. 20
11. 在数据压缩编码中的哈夫曼编码方法, 在本质上是一种 () 的策略。
A. 枚举 B. 贪心 C. 递归 D. 动态规划
12. 由 1,1,2,2,3 这五个数字组成不同的三位数有 () 种。
A. 18 B. 15 C. 12 D. 24
13. 考虑如下递归算法
solve(n)
 if $n \leq 1$ return 1
 else if $n \geq 5$ return $n * \text{solve}(n-2)$
 else return $n * \text{solve}(n-1)$
则调用 solve(7) 得到的返回结果为 ()。
A. 105 B. 840 C. 210 D. 420
14. 以 a 为起点, 对右边的无向图进行深度优先遍历, 则 b,c,d,e 四个点中有可能作为最后一个遍历到的点的个数为 ()。



- A. 1 B. 2 C. 3 D. 4

15.有四个人要从 A 点坐一条船过河到 B 点, 船一开始在 A 点。该船一次最多可坐两个人。已知这四个人中每个人独自坐船的过河时间分别为 1,2,4,8, 且两个人坐船的过河时间为两人独自过河时间的较大者。则最短 () 时间可以让四个人都过河到 B 点 (包括从 B 点把船开回 A 点的时间)。

- A. 14 B. 15 C. 16 D. 17

二、阅读程序 (程序输入不超过数组或字符串定义的范围; 判断题正确填 √ , 错误填 × ; 除特殊说明外, 判断题 1.5 分, 选择题 3 分, 共计 40 分)

16.

```

#include <iostream>
using namespace std;
int n;
int a[1000];
int f(int x)
{
    int ret = 0;
    for(;x;x&=x-1)ret++;
    return ret;
}
14 int g(int x)
15 {
16     return x & -x;
17 }
int main( )
{
    cin>> n;
    for(int i=0;i<n;i++)cin>>a[i];
    for(int i=0;i<n;i++)
        cout<< f(a[i]) + g(a[i]) << ' ';
    cout << endl;
    return 0;
}

```

判断题

- (1) 输入的 n 等于 1001 时, 程序不会发生下标越界。 ()
 (2) 输入的 a[i] 必须全为正整数, 否则程序将陷入死循环。 ()
 (3) 当输入为 5 2 11 9 16 10 时, 输出为 3 4 3 17 5。 ()

(4) 当输入为 1 511998 时, 输出为 18。 ()

(5) 将源代码中 g 函数的定义 (14~ 17 行) 移到 main 函数的后面, 程序可以正常编译运行。 ()

单选题

(6) 当输入为 2 -65536 2147483647 时, 输出为 ()。

A. 65532 33

B. 65552 32

C. 65535 34

D. 65554 33

17.

```
#include <iostream>
#include <string>
using namespace std;
char base[64];
char table[256];
void init()
{
    for(int i=0;i<26;i++)base[i]='A'+i;
    for(int i=0;i<26;i++)base[26+i]='a'+i;
    for(int i=0;i<10;i++)base[52+i]='0'+i;
    base[62] = '+', base[63] = '/';
    for (int i = 0; i < 256; i++) table[i] = 0xff;
    for (int i = 0; i < 64; i++) table[base[i]] = i;
    table['='] = 0;
}
string decode(string str)
{
    string ret; .
    int i;
    for (i = 0; i < str.size(); i+= 4) {
        ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
        if (str[i + 2] != '=')
            ret += (table[str[i + 1]] & 0xf) << 4 | table[str[i + 2]] >> 2;
        if (str[i + 3] != '=')
            ret += table[str[i + 2]] << 6 | table[str[i + 3]];
    }
    return ret;
}
int main()
{
    init();
    cout << int(table[0]) << endl;
    string str;
    cin >> str;
    cout << decode(str) << endl;
    return 0;
}
```

}

判断题

- (1) 输出的第二行一定是由小写字母、大写字母、数字和 +、/、= 构成的字符串。 ()
- (2) 可能存在输入不同，但输出的第二行相同的情形。 ()
- (3) 输出的第一行为 -1。 ()

单选题

- (4) 设输入字符串长度为 n ，decode 函数的时间复杂度为 ()
- A. $O(\sqrt{n})$ B. $O(n)$ C. $O(n \log n)$ D. $O(n^2)$
- (5) 当输入为 Y3Nx 时，输出的第二行为 ()。
- A. csp B. csq C. CSP D. Csp
- (6) (3.5 分) 当输入为 Y2NmIDlwMjE= 时，输出的第二行为 ()。
- A. ccf2021 B. ccf2022 C. ccf 2021 D. ccf 2022

18.

```
#include <iostream>
using namespace std;
const int n = 100000;
const int N=n+1;
int m;
int a[N], b[N], c[N], d[N];
int f[N], g[N];
void init()
{
    f[1]=g[1]=1;
    for(int i=2;i<=n;i++){
        if (!a[i]) {
            b[m++] = i;
            c[i]=1,f[i]=2;
            d[i] = 1, g[i] =i+ 1;
        }
        for(int j=0;j<m&& b[j]*i<=n;j++){
            int k = b[j];
            a[i *k] = 1;
            if(i%k==0) {
                c[i*k]=c[i] + 1;
                f[i*k]=f[i]/c[i*k]*(c[i*k]+1);
                d[i * k] = d[i];
                g[i*k]=g[i]*k+d[i];
                break;
            }
        }
        else {
```

```

        c[i*k]=1;
        f[i*k]=2*f[i];
        d[i * k] = g[i];
        g[i * k]= g[i] * (k + 1);
    }
}
}
int main()
{
    init();
    int x;
    cin>>x;
    cout<< f[x] << ' ' << g[x] << endl;
    return 0;
}

```

假设输入的 x 是不超过 1000 的自然数，完成下面的判断题和单选题：

判断题

- (1)若输入不为 1，把第 13 行删去不会影响输出的结果。（ ）
- (2)(2 分) 第 25 行的 $f[i] / c[i * k]$ 可能存在无法整除而向下取整的情况。（ ）
- (3)(2 分) 在执行完 `init()` 后， f 数组不是单调递增的，但 g 数组是单调递增的。（ ）

单选题

- (4)`init` 函数的时间复杂度为（ ）。
- A. $O(n)$ B. $O(n \log n)$ C. $O(n\sqrt{n})$ D. $O(n^2)$
- (5)在执行完 `init()` 后， $f[1], f[2], f[3] \dots f[100]$ 中有（ ）个等于 2。
- A. 23 B. 24 C. 25 D. 26
- (6)(4 分) 当输入为 1000 时，输出为（ ）。
- A. 15 1340 B. 15 2340 C. 16 2340 D. 16 1340

三、完善程序（单选题，每小题 3 分，共计 30 分）

19.(Josephus 问题) 有 n 个人围成一个圈，依次标号 0 至 $n-1$ 。从 0 号开始，依次 0,1,0,1,... 交替报数，报到 1 的人会离开，直至圈中只剩一个人。求最后剩下人的编号。试补全模拟程序。

```

#include <iostream>
using namespace std;
const int MAXN = 1000000;
int F[MAXN];
int main() {
    int n;
    cin>>n;
    int i=0,p=0, C=0;
    while (①) {
        if (F[i] == 0) {
            if (②) {

```

```

        F[i] = 1;
        ③;
    }
    ④;
}
⑤;
}
int ans = -1;
for(i=0;i<n;i++)
    if (F[i] == 0)
        ans = i;
cout << ans << endl;
return 0;
}

```

(1)①处应填 ()

A.i < n B.c < n C.i < n- 1 D.c < n-1

(2)②处应填 ()

A.i % 2 == 0 B.i % 2 == 1 C.p D.!p

(3)③处应填 ()

A.i++ B.i = (i + 1) % n C.c++ D.p ^= 1

(4)④处应填 ()

A.i++ B.i = (i + 1) % n C.c++ D.p ^= 1

(5)⑤处应填 ()

A.i++ B.i = (i + 1) % n C.c++ D.p ^= 1

20. (矩形计数) 平面上有 n 个关键点, 求有多少个四条边都和 x 轴或者 y 轴平行的矩形, 满足四个顶点都是关键点。给出的关键点可能有重复, 但完全重合的矩形只计一次。

```

#include <iostream>
using namespace std;
struct point {
    int x, y, id;
};
bool equals(point a, point b) {
    return a.x == b.x && a.y == b.y;
}
bool cmp(point a, point b) {
    return ①;
}
void sort(point A[], int n) {
    for(int i=0;i<n;i++)
        for(int j=1;j<n;j++)
            if (cmp(A[j], A[j - 1])) {

```

```

        point t = A[j];
        A[j] = A[j - 1];
        A[j-1]=t;
    }
}

int unique(point A[], int n) {
    int t=0;
    for(int i=0;i<n;i++)
        if (②)
            A[t++] = A[i];
    return t;
}

bool binary_search(point A[], int n, int x, int y) {
    point p;
    p.x = x;
    p.y = y;
    p.id = n;
    int a = 0, b = n - 1;
    while(a < b) {
        int mid = ③;
        if (④)
            a = mid + 1;
        else
            b = mid;
    }
    return equals(A[a], p);
}

const int MAXN = 1000;
point A[MAXN];
int main() {
    int n;
    cin >> n;
    for(int i=0; i<n; i++){
        cin >> A[i].x >> A[i].y;
        A[i].id = i;
    }
    sort(A, n);
    n = unique(A, n);
    int ans = 0;
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            if (⑤ && binary_search(A, n, A[i].x, A[j].y) &&
                binary_search(A, n, A[j].x, A[i].y)) {
                ans++;
            }
}

```

```

    }
    cout << ans << endl;
    return 0;
}

```

试补全枚举算法。

(1)①处应填 ()

- A. $a.x \neq b.x ? a.x < b.x : a.id < b.id$ B. $a.x \neq b.x ? a.x < b.x : a.y < b.y$
 C. $\text{equals}(a, b) ? a.id < b.id : a.x < b.x$
 D. $\text{equals}(a, b) ? a.id < b.id : (a.x \neq b.x ? a.x < b.x : a.y < b.y)$

(2)②处应填 ()

- A. $i == 0 \parallel \text{cmp}(A[i], A[i - 1])$ B. $t == 0 \parallel \text{equals}(A[i], A[t - 1])$
 C. $i == 0 \parallel !\text{cmp}(A[i], A[i - 1])$ D. $t == 0 \parallel !\text{equals}(A[i], A[t - 1])$

(3)③处应填 ()

- A. $b - (b - a) / 2 + 1$ B. $a + b + 1 \gg 1$
 C. $(a + b) \gg 1$ D. $a + (b - a + 1) / 2$

(4)④处应填 ()

- A. $!\text{cmp}(A[\text{mid}], p)$ B. $\text{cmp}(A[\text{mid}], p)$
 C. $\text{cmp}(p, A[\text{mid}])$ D. $!\text{cmp}(p, A[\text{mid}])$

(5)⑤处应填 ()

- A. $A[i].x == A[j].x$ B. $A[i].id < A[j].id$
 C. $A[i].x == A[j].x \ \&\& \ A[i].id < A[j].id$ D. $A[i].x < A[j].x \ \&\& \ A[i].y < A[j].y$