讲评

TEACHING COURSEWARE POWPOINT

授课时间：2025.08.08

目录

01

选择

TEACHING
COURSEWARE

TEACH

选择

1.二进制数
11 1011 1001 0111 和 01 0110 1110 1011 进行按位与运算的结果是（）
A. 01 0010 1000 1011
B. 01 0010 1001 0011
C. 01 0010 1000 0001
D. 01 0010 1000 0011
2.一个 32 位整型变量占用（）个字节。
A. 32　　B. 128　　C. 4　　D. 8
3.若有如下程序段，其中 s、a、b、c 均已定义为整型变量,且 a、c 均已赋值（c 大于 0）
s = a;
for （b = 1; b <= c; b++） s = s - 1;
则与上述程序段功能等价的赋值语句是（）
A. s = a - c;　　B. s = a - b;
C. s = s - c;　　D. s = b - c;
4.设有 100 个已排好序的数据元素，采用折半查找时，最大比较次数为（）
A. 7　　B. 10
C. 6　　D. 8
5.新学期开学了，小胖想减肥，健身教练给小胖制定了两个训练方案。
方案一：每次连续跑 3 公里可以消耗 300 千卡（耗时半小时）；
方案二：每次连续跑 5 公里可以消耗 600 千卡（耗时 1 小时）。
小胖每周周一到周四能抽出半小时跑步，周五到周日能抽出一小时跑步。
另外，教练建议小胖每周最多跑 21 公里，否则会损伤膝盖。
请问如果小胖想严格执行教练的训练方案，并且不想损伤膝盖，每周最多通过跑步消耗多少千卡？（）
A. 3000
B. 2500
C. 2400
D. 2520

基础常识：1 字节 = 8 位，32 位整型变量占用 32÷8=4 字节

选择

6.以下哪个奖项是计算机科学领域的最高奖？（）

A. 图灵奖

B. 鲁班奖

C. 诺贝尔奖

D. 普利策奖

7.排序的算法很多，若按排序的稳定性和不稳定性分类，则（）是不稳定排序。

A. 冒泡排序

B. 插入排序

C. 选择排序

D. 计数排序

8.在内存储器中每个存储单元都被赋予一个唯一的序号，称为（）。

A. 地址

B. 序号

C. 下标

D. 编号

9.编译器的主要功能是（ ）。

A. 将源程序翻译成机器指令代码

B. 将源程序重新组合

C. 将低级语言翻译成高级语言

D. 将一种高级语言翻译成另一种高级语言

选择

10.冒泡排序算法的伪代码如下：

```
输入：数组 L，n ≥ k。输出：按非递减顺序排序的 L。
算法 BubbleSort：
   1. FLAG ← n //标记被交换的最后元素位置
   2. while FLAG > 1 do
   3.     k ← FLAG -1
   4.     FLAG ← 1
   5.     for j=1 to k do
   6.         if L(j) > L(j+1) then do
   7.             L(j)  ↔ L(j+1)
   8.             FLAG ← j
```

对 n 个数用以上冒泡排序算法进行排序，最少需要比较多少次？（ ）。

A. n^2　　B. n-2　　C. n-1　　D. n

11.设 A 是 n 个实数的数组，考虑下面的递归算法：

```
XYZ (A[1..n])
1.  if n=1 then return A[1]
2.  else temp ← XYZ (A[1..n-1])
3.  if temp < A[n]
4.  then return temp
5.  else return A[n]
```

请问算法 XYZ 的输出是什么？（）。

A. A 数组的平均
B. A 数组的最小值
C. A 数组的中值
D. A 数组的最大值

10.冒泡排序最少比较次数
最优情况：数组已有序，仅需1 轮比较（n-1 次），选 C。
11.递归算法功能
逻辑：递归比较前n-1 个元素的最小值与第 n 个元素，返回更小值，即求数组最小值，选 B。

选择

12.二进制数 1011 转换成十进制数是（ ）。

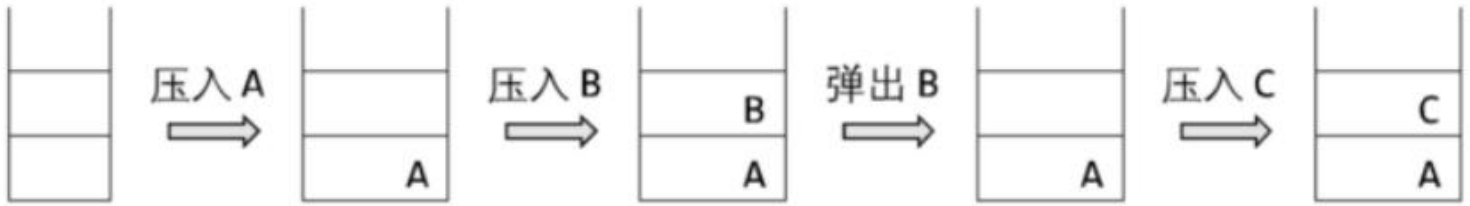
A. 11

B. 10

C. 13

D. 12

13.下图中所使用的数据结构是（)。



A. 栈　　B. 队列　　C. 二叉树　　D. 哈希表

14.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

15.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

02

阅读程序

TEACHING
COURSEWARE

TEACH

阅读程序


```
#include <cstdio>
#include <cstring>
using namespace std;
char st[100];
int main() {
    scanf("%s", st);
    int n = strlen(st);
    for (int i = 1; i <= n; ++i) {
        if (n % i == 0) {
            char c = st[i - 1];
            if (c >= 'a')
                st[i - 1] = c - 'a' + 'A';
        }
    }
    printf("%s", st);
    return 0;
}
```

阅读程序

判断题

1.输入的字符串只能由小写字母或大写字母组成。（）

2.若将第 8 行的 i = 1 改为 i = 0，程序运行时会发生错误。（）

3.若将第 8 行的 i <= n 改为 i * i <= n，程序运行结果不会改变。（）

4.若输入的字符串全部由大写字母组成，那么输出的字符串就跟输入的字符串一样。（）

选择题

5.若输入的字符串长度为 18，那么输入的字符串跟输出的字符串相比，至多有（）个字符不同。

6.若输入的字符串长度为（），那么输入的字符串跟输出的字符串相比，至多有 36 个字符不同。

1. A. 正确 B. 错误

2. A. 正确 B. 错误

3. A. 正确 B. 错误

4. A. 正确 B. 错误

5. A. 18 B. 6 C. 10 D. 1

6. A. 36 B. 100000 C. 1 D. 128

阅读程序

判断题1：输入的字符串只能由小写字母或大写字母组成。
解析：程序未限制输入字符类型（可包含数字、符号等），仅对小写字母处理。输入其他字符时程序仍能运行，故说法错误。
答案：B
判断题2：若将第 8 行的i = 1改为i = 0，程序运行时会发生错误。
解析：i=0时，循环条件i <= n成立，此时n % i即 “除以 0”，属于运行时错误（除数为 0）。故说法正确。
答案：A
判断题3：若将第 8 行的i <= n改为i * i <= n，程序运行结果不会改变。
解析：原逻辑需处理n的所有约数（如n=6的约数为 1,2,3,6），而i*i <= n仅能找到小于等于√n的约数（如n=6仅能找到 1,2），漏掉大于√n的约数（3,6），导致处理位置减少，结果改变。故说法错误。
答案：B
判断题4：若输入的字符串全部由大写字母组成，输出与输入一致。
解析：程序仅对小写字母（c >= 'a'）进行转换，大写字母不满足条件，保持不变。故说法正确。
答案：A
选择题5：字符串长度为 18 时，至多有多少个字符不同？
解析：不同字符数取决于n=18的约数个数（每个约数对应位置可能被转换）。18 的约数为 1,2,3,6,9,18，共 6 个，故最多 6 个字符不同。
答案：B
选择题6：字符串长度为多少时，至多有 36 个字符不同？
解析：36 个不同字符对应 36 个约数。一个数的约数个数由其质因数分解决定（如100000=2^5×5^5，约数个数为(5+1)×(5+1)=36），故长度为 100000 时满足。
答案：B

阅读程序

```
#include<cstdio>
using namespace std;
int n, m;
int a[100], b[100];

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i)
        a[i] = b[i] = 0;
    for (int i = 1; i <= m; ++i) {
        int x, y;
        scanf("%d%d", &x, &y);
        if (a[x] < y && b[y] < x) {
            if (a[x] > 0)
                b[a[x]] = 0;
            if (b[y] > 0)
                a[b[y]] = 0;
            a[x] = y;
            b[y] = x;
        }
    }
    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        if (a[i] == 0)
            ++ans;
        if (b[i] == 0)
            ++ans;
    }
    printf("%d", ans);
    return 0;
}
```

阅读程序

假设输入的 n 和 m 都是正整数，x 和 y 都是在 [1,n] 的范围内的整数，完成

下面的判断题和单选题：

判断题

1.当 m>0 时，输出的值一定小于 2n。（）

2.执行完第 27 行的 ++ans 时，ans 一定是偶数。（）

3.a[i] 和 b[i] 不可能同时大于 0。（）

4.右程序执行到第 13 行时，x 总是小于 y，那么第 15 行不会被执行。（）

选择题

5.若 m 个 x 两两不同，且 m 个 y 两两不同，则输出的值为（）

6.若 m 个 x 两两不同，且 m 个 y 都相等，则输出的值为（）

1. A. 正确 B. 错误

2. A. 正确 B. 错误

3. A. 正确 B. 错误

4. A. 正确 B. 错误

5.A. 2n−2m B. 2n+2 C. 2n−2 D. 2n

6.A. 2n−2 B. 2n C. 2m D. 2n−2m

阅读程序

这个程序是处理一些“成对的数字”（比如x和y），记录哪些x和y建立了关联，最后算一算有多少个数字既没和别人建立关联，也没被别人关联。
看变量
n：所有数字的范围（比如数字都是 1 到n之间的）。
m：要处理的 “成对数字” 有m组（比如第 1 组是x1,y1，第 2 组是x2,y2）。
a[x]：记录 “数字x当前和哪个数字关联着”（如果是 0，说明x没关联任何数字）。
b[y]：记录 “数字y当前被哪个数字关联着”（如果是 0，说明y没被任何数字关联）。
ans：最后要输出的结果，就是 “没关联的数字” 总共有多少个。
初始化：一开始所有数字都“没关联”（a和b全是 0）。
处理每组 x 和 y：
只有新的y比x之前关联的y'大，并且新的x比y之前被关联的x'大，才会更新关联。
更新时，要先取消x和y之前的旧关联，再建立新关联。
统计结果：把“没关联任何数字的a[i]” 和 “没被任何数字关联的b[i]”数量加起来，就是答案。

阅读程序

判断题 1：当m>0时，输出的值一定小于2n。
解析：总共有2n个 “关联位置”（a[1..n]记录n个数字的关联，b[1..n]记录n个数字被关联）。
当m>0时，至少有 1 组有效关联，意味着a和b中各至少有 1 个非 0 值（即至少 2 个位置已关联）。
因此，未关联的数量最多为2n - 2，必然小于2n。
结论：正确（选 A）。
判断题 2：执行完第 27 行的++ans时，ans一定是偶数。
解析：第 27 行是 “若a[i]==0，ans加 1”（统计未关联的a[i]）。
此时ans只加了 1 次，可能是奇数（比如第一次执行时ans从 0 变成 1）。
结论：错误（选 B）。

阅读程序

判断题 3：a[i]和b[i]不可能同时大于 0。
解析：a[i]表示 “数字i关联的数字”，b[i]表示 “数字i被哪个数字关联”，两者描述的是i的不同角色（主动关联和被关联）。
例如：a[2]=3（数字 2 关联 3）和b[2]=1（数字 2 被 1 关联），此时两者都大于 0。
结论：错误（选 B）。
判断题 4：若x总是小于y，第 15 行不会被执行。
解析：第 15 行（b[a[x]]=0）的作用是 “取消x之前的关联”，与x和y的大小无关。
只要x之前有关联（a[x]>0），无论x和y谁大，都会执行该行。
结论：错误（选 B）。
选择题 5：若m个x两两不同，且m个y两两不同，输出的值为？
解析：m个x和m个y均无重复，说明所有关联都有效（没有冲突）。
此时a中有m个非 0 值（n-m个未关联），b中有m个非 0 值（n-m个未关联）。
总未关联数量：(n-m)+(n-m)=2n-2m。
结论：选 A（2n−2m）。
选择题 6：若m个x两两不同，且m个y都相等，输出的值为？
解析：y固定为同一个值（如y0），根据关联规则，y0只会保留与 “最大的x” 的关联（其他x因不满足 “更优” 被淘汰）。
最终只有 1 组有效关联，因此a中有n-1个未关联，b中有n-1个未关联。
总未关联数量：(n-1)+(n-1)=2n-2。
结论：选 A（2n−2）。

阅读程序

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

long long n, ans;
int k, len;
long long d[1000000];

int main() {
  cin >> n >> k;
  d[0] = 0;
  len= 1;
  ans = 0;
  for (long long i = 0; i <n; ++i) {
    ++d[0];
    for (int j = 0; j + 1<len; ++j) {
      if (d[j] == k) {
        d[j] = 0;
        d[j + 1] += 1;
        ++ans;
      }
    }
    if (d[len- 1] == k) {
      d[len - 1] = 0;
      d[len] =1;
      ++len;
      ++ans;
    }
  }
  cout << ans << endl;
  return 0;
}
```

阅读程序

假设输入的 n 是不超过 2^62 的正整数，k 都是不超过 10000 的正整数，完成下面的判断题和单选题：

判断题

1.若 k=1，则输出 ans 时，len=n。（ ）

2.若 k>1，则输出 ans 时，len 一定小于 n。（ ）

3.若 k>1，则输出 ans 时，k^len一定大于 n。（ ）

单选题

4.若输入的 n 等于：10^15 ，输入的 k 为 1 则输出等于（ ）。

5.若输入的 n 等于 205,891,132,094,649（即 3^{30}），输入的 k 为 3，则输出等于（ ）

6.若输入的 n 等于 100,010,002,000,090，输入的 k 为 10，则输出等于（ ）。

1. A. 正确 B. 错误

2.A. 正确 B. 错误

3.A. 正确 B. 错误

4. A. 1　　B. $(10^{30} - 10^{15})/2$

C. $(10^{30} + 10^{15})/2$　　D. 10^{15}

5.

A. 3^{30}　　B. $(3^{30} - 1)/2$

C. $3^{30} - 1$　　D. $(3^{30} + 1)/2$

6.

A. 11,112,222,444,543

B. 11,122,222,444,453

C. 11,122,222,444,543

D. 11,112,222,444,453

阅读程序

判断题1：当n=1时：
初始：d[0]=0，len=1，ans=0。
第 1 次加 1：d[0]变为 1（满 1），触发进位：
d[0]清零（d[0]=0），新增高位d[1]=1，len变为 2。
ans加 1（ans=1）。
结果：len=2，n=1 → len≠n。
当n=2时：
第 2 次加 1：d[0]变为 1（满 1），触发进位：
d[0]清零，d[1]加 1（d[1]=2，不满 1，无需再进位）。
ans加 1（ans=2）。
结果：len=2（高位未再增加），n=2 → 此时len=n（巧合）。
当n=3时：
第 3 次加 1：d[0]变为 1（满 1），触发进位：
d[0]清零，d[1]加 1（d[1]=3，仍不满 1）。
ans加 1（ans=3）。
结果：len=2，n=3 → len≠n。
当n=4时：
第 4 次加 1：d[0]变为 1（满 1），触发进位：
d[0]清零，d[1]加 1（d[1]=4，不满 1）。
ans加 1（ans=4）。

阅读程序

判断题2：若k>1，len一定小于n?
反例：k=2（二进制），n=1：
数1 次：d[0] = 1（不满 2），len=1，n=1，此时len = n，不小于。
结论：错误（选B）。
判断题3：若k>1，k^len一定大于n?
解释：len是当前的位数，k^len是 “len+1 位数的最小值”（比如 2 位数的十进制，10^2=100是最小的 3 位数）。而我们数到的最大数是n，一定小于最小的len+1位数（比如 2 位数最大是 99 < 100）。
例子：k=10，len=2（2 位数），10^2=100，而 n 最大是 99 < 100。
结论：正确（选A）。
选择题4：n=10^15，k=1，输出是?
解释：k=1时，每次加 1 必进位（因为 1 进制下 1 是最大），n 次加 1 就进位 n 次。
结论：10^15（选 D）。

03

完善程序

TEACHING
COURSEWARE

TEACH

完善程序

输入月份 m(1≤m≤12)，按一定格式打印 2015 年第 m 月的月历。（第三、四空 2.5 分， 其余 3 分）

例如，2015 年 1 月的月历打印效果如下（第一列为周日）：

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

完善程序

```
#include <iostream>
#include <string>
using namespace std;
const int dayNum[] = {-1, 31, 28, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31};
int m, offset, i;
int main(){
    cin >> m;
    cout << "S\tM\tT\tW\tT\tF\tS" << endl; /* '\t'为 TAB 制表
符 */
    ①;
    for (i = 1; i < m; i++)
        offset = ②;
    for (i = 0; i < offset; i++)
        cout << '\t';
    for (i = 1; i <= ③; i++)
    {
        cout << ④;
        if (i == dayNum[m] || ⑤ == 0)
            cout << endl;
        else
            cout << '\t';
    }
    return (0);
}
```

完善程序

1. ①处应填（ ）

A.	offset = 0	B.	offset = 4
C.	offset = 6	D.	offset = 1

2. ②处应填（ ）

A.	offset + dayNum[i]	B.	(offset + dayNum[m]) % 7
C.	(offset + dayNum[i]) % 7	D.	dayNum[i] % 7

3. ③处应填（ ）

A.	dayNum[m-1]	B.	dayNum[m]
C.	m	D.	31

4. ④处应填（ ）

A.	offset	B.	dayNum[i]
C.	i+1	D.	i

5. ⑤处应填（ ）

A.	(offset + i) % 7	B.	(offset + i - 1) % 7
C.	i % 7	D.	offset % 7

完善程序

1.①处答案：offset = 4

解析：2015 年 1 月 1 日是周四，表头第一列是周日，所以 1 月 1 日前面有 4 个空白（周四在第 5 个位置），初始偏移量为 4。

1.②处答案：(offset + dayNum[i]) % 7

解析：每月 1 日的偏移量 = 上月最后一天的偏移量 + 上月天数，对 7 取模（每周 7 天循环）。例如 2 月 1 日的偏移 = （1 月偏移 4 + 31 天）%7 = （4+3）%7=0（因为 31%7=3）。

1.③处答案：dayNum[m]

解析：dayNum[m]存储当月总天数（如 3 月对应 31），循环需遍历 1 到当月天数。

1.④处答案：i

解析：i是当前日期（1 到当月天数），直接打印即可。

1.⑤处答案：(offset + i) % 7

解析：(offset + i) %7表示第 i 天在周几（0 = 周日，6 = 周六）。若为 0，说明是周六，需要换行；若到月末（i = 当月天数），也需换行。

完善程序

有 n 名同学参加学校组织的郊游活动，已知学校给这 n 名同学的郊游总经费为 A 元，与此同时第 i 位同学自己携带了 Mi 元。为了方便郊游，活动地点提供 B(>=n) 辆自行车供人租用，租用第 j 辆自行车的价格为 Cj 元，每位同学可以使用自己携带的钱或者学校的郊游经费，为了方便账务管理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多有多少位同学能够租用到自行车。

本题采用二分法。对于区间[l, r]，我们取中间点 mid 并判断租用到自行车的人数能否达到 mid。判断的过程是利用贪心算法实现的。

完善程序

```
01 #include <iostream>
02 using namespace std;
03 #define MAXN 1000000
04
05 int n, B, A, M[MAXN], C[MAXN], l, r, ans, mid;
06
07 bool check(int nn)
08 {
09     int count = 0, i, j;
10     i = ①;
11     j = 1;
12     while (i <= n)
13     {
14         if (②)
15             count += C[j] - M[i];
16         i++;
17         j++;
18     }
19     return ③;
20 }
21
```

完善程序

```
22 void sort(int a[], int l, int r)
23 {
24     int i = l, j = r, x = a[(l + r) / 2], y;
25     while (i <= j)
26     {
27         while (a[i] < x) i++;
28         while (a[j] > x) j--;
29         if (i <= j)
30         {
31             y = a[i];
32             a[i] = a[j];
33             a[j] = y;
34             i++;
35             j--;
36         }
37     }
38     if (i < r) sort(a, i, r);
39     if (l < j) sort(a, l, j);
40 }
```

完善程序

```
42 int main()
43 {
44     int i;
45     cin >> n >> B >> A;
46     for (i = 1; i <= n; i++)
47         cin >> M[i];
48     for (i = 1; i <= B; i++)
49         cin >> C[i];
50     sort(M, 1, n);
51     sort(C, 1, B);
52     l = 0;
53     r = n;
54     while (l <= r)
55     {
56         mid = (l + r) / 2;
57         if (④)
58         {
59             ans = mid;
60             l = mid + 1;
61         }
62         else
63             r = ⑤;
64     }
65     cout << ans << endl;
66     return 0;
67 }
```

完善程序

1. ①处应填（B）

A.	1	B.	n - nn + 1
C.	nn	D.	n - nn

2. ②处应填（C）

A.	M[i] == C[j]	B.	M[i] > C[j]
C.	M[i] < C[j]	D.	M[i] != C[j]

3. ③处应填（B）

A.	count >= A	B.	count <= A
C.	count < A	D.	count > A

4. ④处应填（A）

A.	check(mid)	B.	!check(mid)
C.	mid <= A	D.	mid >= A

5. ⑤处应填（D）

A.	1	B.	mid
C.	mid + 1	D.	mid - 1

完善程序

1.①处答案：n - nn + 1

解析：M升序排序后，M[n]是最大的（自带钱最多）。选自带钱最多的nn名同学，起始索引为n - nn + 1（如 nn=2，选 M [n-1] 和 M [n]）。

1.②处答案：M[i] < C[j]

解析：若同学自带钱M[i]小于车价C[j]，学校需补差价C[j] - M[i]；若足够，则无需补钱。

1.③处答案：count <= A

解析：总差价count需≤学校经费A，才能满足nn名同学租车。

1.④处答案：check(mid)

解析：二分法中，若check(mid)返回 true（能租给 mid 名同学），则尝试更多人（更新ans，l=mid+1）。

1.⑤处答案：mid - 1

解析：若check(mid)返回 false（不能满足），则减少人数，调整右边界r=mid-1。