选择题答案

1. A
2. D
3. C
4. A
5. A
6. D
7. C
8. C
9. B
10. C
11. C
12. A
13. C 第1, 2位有5种选法（0,1,6,8,9),第3位有3种(0,1,8),第4, 5位由前2位决定，答案为5*5*3=75
14. B
15. A

二、阅读程序

1.

```
#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++) {  //第8行
        if (n % i == 0) {
            char c = st[i - 1];
            if (c >= 'a')
                st[i - 1] = c - 'a' + 'A';
        }
    }
    printf("%s", st);
}
```

判断题

1）输入的字符串只能由小写字母或大写字母组成。（错）

2）若将第8行的“i = 1”改成“i= 0”，程序运行时会发生错误。 （对）

3）若将第8行的“i <= n”改成“i * i <= n”，程序运行结果不会改变。 （错，参考第5题）

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。 （对）

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（B）个字符不同

A. 18　B. 6　C. 10　D. 1
分析：其实就是从1到18的数字中，一共有哪些数字被18整除？有1，2，3，6，9，18，一共6个

6）若输入的字符串长度为（B），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。
A. 36　B. 100000　C. 1　D. 128
分析：排除法，1到128的数字中，能被128整除的数字达不到36个（1，2，4，8，16，。。。），所以只能选B。

2.

```
#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) {    //第13行

            if (a[x] > 0)
                b[a[x]] = 0;          //第15行

            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\n", ans);
}
```

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。

判断题

1）当m>0时，输出的值一定小于2n。（对）

2）执行完第27行的“++ans”时，ans一定是偶数。 （错）

3）a[i]和b[i]不可能同时大于0。 （错）

4）若程序执行到第13行时，x总是小于y，那么第15行不会被执行。（错，假设输入n=m=3，然后依次输入x=1,y=2和x=1,y=3）

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 A（假设输入n=m=3，然后依次输入x=1,y=2和x=2,y=3和x=3,y=1，最后结果ans=0；假设输入n=3,m=2，然后依次输入x=1,y=2和x=2,y=3，最后结果a[1] = b[1] = 0，所以ans=2；）

A. 2n-2m B. 2n+2 C. 2n-2 D. 2n

6）若m个x两两不同，且m个y都相等，则输出的值为 A（假设输入n=m=3，然后依次输入x=1,y=2和x=2,y=2和x=3,y=2，最后结果a[1] = 0，a[2] = 0，a[3] = 2，b[1] = b[3] = 0，b[2] = 3，所以ans=4；）

A. 2n-2 B. 2n C. 2m D. 2n-2m

分析：这道题很难理解，我们需要会用假设。

3.

```
#include <iostream>
using namespace std;
const int maxn = 10000;
int n;
int a[maxn];
int b[maxn];
int f(int l, int r, int depth) {
    if (l > r)
        return 0;
    int min = maxn, mink;
    for (int i = l; i <= r; i++) {
        if (min > a[i]) {
            min = a[i];
            mink = i;
        }
    }
    int lres = f(l, mink - 1, depth + 1);
    int rres = f(mink + 1, r, depth + 1);
    return lres + rres + depth * b[mink];
}
int main()
{
    cin >> n;
```

```
    for (int i  = 0; i < n; i++)
        cin >> a[i];
    for (int i = 0; i < n; i++)
        cin >> b[i];
    cout << f(0, n - 1, 1) << endl;
    return 0;
}
```

判断题

1）如果a数组有重复的数字，则程序运行时会发生错误。（错）

2）如果b数组全为0，则输出为0。 （对）

选择题

3）当n=100时，最坏情况下，与第12行的比较运算执行的次数最接近的是 A

A. 5000 B. 600 C. 6 D. 100

分析：最坏情况下，也就是这颗二叉树的高度尽可能的深，所以这棵树只有右子节点，高度就是100，从根节点开始，每一层的比较执行次数分别是100，99，98，...,1。累加的结果5050

4）当n=100时，最好情况下，与第12行的比较运算执行的次数最接近的是 D

A. 100 B. 6 C. 5000 D. 600

分析：最好情况下，二叉树的高度尽可能的矮，高度是log(100)向下取整+1=6，从根节点开始，每一层的比较执行次数分别是100，99，97，93，85，69。累加的结果最接近600

5）当n=10时，若b数组满足，对任意0<=i<n，都有b[i] = i + 1，那么输出最大为 D

A. 386 B. 383 C. 384 D. 385

分析：可以画一颗二叉树来表示递归函数，树的高度要尽可能深（这样depth会比较大），所以这棵树只有右子节点。这棵树的深度是10。输出1*1 + 2*2 + 3*3 + ... + 10*10 = 385

6）当n=100时，若b数组满足，对任意0<=i<n，都有b[i] = 1，那么输出最小为

A. 582 B. 580 C. 579 D. 581

分析：要输出最小，所以二叉树的高度尽量矮，是一颗完全二叉树。

1*1+2*2+4*3+8*4+16*5+32*6+37*7=580

三、完善程序

1. （矩阵变幻）有一个奇幻的矩阵，在不停的变幻，其变幻方式为：数字0变成矩阵：

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

，数字1变成矩阵：

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{4}$$

最初该矩阵只有一个元素0，变幻n次后，矩阵会变成什么样?

例如，矩阵最初为[0]，变幻2次后：

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{4}$$

输入一行一个不超过10的正整数n，输出变幻n次后的矩阵。

试补全程序。

提示："<<"表示二进制左移运算符，例如(11) << 2 = (1100)，"^"表示二进制异或运算符，它将两个参与运算的数中的每个对应的二进制位一一进行比较，若两个二进制位相同，则运算结果对应二进制位为0，反之为1.

```
#include <cstdio>
using namespace std;
int n;
const int max_size = 1 << 10;

int res[max_size][max_size];

void recursive(int x, int y, int n, int t) {
  if (n == 0) {
    res[x][y] = (1);
    return;
  }
  int step = 1 << (n - 1);
  recursive((2), n - 1, t);  // 应该就是 (x,y)坐标本身
  recursive(x, y + step, n - 1, t);  //  (x,y) 的下边
  recursive(x + step, y, n - 1, t);  // (x,y)的右边位置
  recursive((3), n - 1, !t);    //根据!t，猜出是(x,y)坐标的右下位置
}

int main() {
  scanf("%d", &n);
  recursive(0, 0, (4));
  int size = (5);
  for (int i = 0; i < size; ++i) {
    for (int j = 0; j < size; ++j)
      printf("%d", res[i][j]);
    puts("");
  }
  return 0;
}

1) (1) 处应填 C  (A和B是一样的)
A. n%2    B. 0    C. t    D. 1

2) (2) 处应填 D
A. x - step, y - step   B. x, y - step
C. x - step, y          D. x, y
```

3）（3）处应填 B

A. x - step, y - step　　B. x + step, y + step

C. x - step, y　　D. x, y - step

4）（4）处应填 B （假设输入n为1，表示变幻一次，那么下面包含n-1的答案肯定都是错的，因为会立刻触发recursive函数的退出条件。）

A. n - 1, n%2　　B. n, 0

C. n, n%2　　D. n - 1, 0

5）（5）处应填 B （假设输入n为1，也就是变幻一次，得到矩阵的边长是2，下面只有1 << n得到2，也就是二进制1左移一位。）

A. 1 << (n + 1)　　B. 1 << n

C. n + 1　　D. 1 << (n - 1)

2. (计数排序) 下面的程序使用双关键字计数排序，将n对10000以内的整数，从小到大排序。例如有三对整数（3，4）（2，4）（3，3），那么排序之后应该是（2，4）（3，3）（3，4）

   输入第一行为n，接下来n行，第i行有两个数a[i]和b[i]，分别表示第i对整数的第一关键字和第二关键字。

   提示：应先对第二关键字排序，再对第一关键字排序，数组ord[]存储第二关键字排序的结果，数组res[]存储双关键字排序的结果。

   试补全程序。

```
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 10000000;
const int maxs = 10000;

int n;
unsigned a[maxn], b[maxn], res[maxn], ord[maxn];  //数组a[]存放第一关键字，b[]存放第二关键字
unsigned cnt[maxs + 1];

int main() {
  scanf("%d", &n);
  for (int i = 0; i < n; ++i)
    scanf("%d%d", &a[i], &b[i]);
  memset(cnt, 0, sizeof(cnt));  //数组cnt[]全部元素的值设置为0
  for (int i = 0; i < n; ++i)
    (1); //利用cnt数组统计数量
  for (int i = 0; i < maxs; ++i)
    cnt[i + 1] += cnt[i];
  for (int i = 0; i < n; ++i)
    (2); //记录初步排序结果
```

```
  memset(cnt, 0, sizeof(cnt));
  for (int i = 0; i < n; ++i)
    (3); //利用cnt数组统计数量
  for (int i = 0; i < maxs; ++i)
    cnt[i + 1] += cnt[i];
  for (int i = n - 1; i >= 0; --i)
    (4); //记录最终排序结果
  for (int i = 0; i < n; ++i)
    printf("%d %d\n", (5));
  return 0;
}
```

1) (1) 处应填 B
  A. ++cnt[i]
  B. ++cnt[b[i]]
  C. ++cnt[a[i]] * maxs + b[i]
  D. ++cnt[a[i]]

2) (2) 处应填 D
  A. ord[--cnt[a[i]]] = i
  B. ord[--cnt[b[i]]] = a[i]
  C. ord[--cnt[a[i]]] = b[i]
  D. ord[--cnt[b[i]]] = i

3) (3) 处应填 C
  A. ++cnt[b[i]]
  B. ++cnt[a[i]] * maxs + b[i]
  C. ++cnt[a[i]]
  D. ++cnt[i]

4) (4) 处应填 A
  A. res[--cnt[a[ord[i]]]] = ord[i]
  B. res[--cnt[b[ord[i]]]] = ord[i]
  C. res[--cnt[b[i]]] = ord[i]
  D. res[--cnt[a[i]]] = ord[i]

5) (5) 处应填 B
  A. a[i], b[i]
  B. a[res[i]], b[res[i]]
  C. a[ord[res[i]]], b[ord[res[i]]]
  D. a[res[ord[i]]], b[res[ord[i]]]

分析：基数排序的动态图，跟这道题比较类似，数组cnt[]就类似是动态图的下面的0到9的桶，第一次是根据个位数比较（也就是题目中的第二关键字），结果存放在数组ord[]中（动态图中是把数字从0到9的桶中放回到原数组），所以这里的区别是，数组ord[]只保存了元素在原数组中的位置，例如（3，4）（2，4）（3，3）三对数字（下标对应为0，1，2），ord[]中依次存放的是2、1、0。然后再根据第一关键字比较，最终结果放到res数组（1，2，0）