第十三届趣味C预赛 Day1 Solution by MSC@ZJU

基本信息

- **预计难度：** 7-1 < 7-2 < 7-3 < 7-4 < 7-5
- **实际通过人数：** 7-1 < 7-2 < 7-4 < 7-3 < 7-5
- **提交&通过情况**
  -

标号	标题	分数	通过数	提交数	通过率
7-1	决赛日期	100	353	580	0.61
7-2	是秘密啊	100	204	1841	0.11
7-3	玩球	100	185	805	0.23
7-4	计算题	100	235	1140	0.21
7-5	Data structure alignment	100	106	448	0.24

是秘密啊

题目大意

给定一个日期，输出对应的星座。

解题思路

所使用的版本来自百度百科：https://baike.baidu.com/item/十二星座/71359

标程

```c
#include <stdio.h>
const char ch[12][50] =
{"Capricorn","Aquarius","Pisces","Aries","Taurus","Gemini",
"Cancer","Leo","Virgo","Libra","Scorpio","Sagittarius"};
const int d[13] = {0,19,18,20,19,20,21,22,22,22,23,22,21};
int n, m;

void solve() {
    scanf("%d%d", &n, &m);
    int ans = 0;
    if ((n==1 && m<=19) || (n==12 && m>=22)) return puts(ch[0]), (void)0;
    for (int _=1;_<=12;_++) if (n>_ || (n==_ && m>d[_]) ) ans++;
    puts(ch[ans]);
```

```
}

int main() {
    int T; scanf("%d",&T);
    while (T--) solve();
    return 0;
}
```

玩球

题目大意

桌面上有编号为1,2,3的三个盒子，每个盒子里面有一个球。 1号盒子里的球是红球，2、3号盒子里的球是白球。 现在有M个操作，每次从x_i 盒子中取任意一个球放入y_i盒子。 现在询问每个盒子在所有操作结束后里面是否有可能有红球。

解题思路

我们称可能有红球的盒子是红色的。 对于一次操作 1、y_i盒子颜色的变动 如果x_i不是红色的，那么这次操作不影响y_i的颜色。 如果x_i是红色的，那么y_i盒子也是红色。 2、x_i盒子颜色的变动 如果x_i盒子在操作后没有球了，那么该盒子一定不可能是红色。 如果x_i盒子在操作后还有球，则该盒子颜色不变。

标程

```
#include <stdio.h>
#include <string.h>
inline int rd() {int r;scanf("%d",&r);return r;}
int a[5], b[5], n, T;
int main() {
    for (int T=rd();T;T--) {
        n = rd();
        a[1] = a[2] = a[3] = 1;
        b[1] = 1, b[2] = b[3] = 0;
        for (int i=1;i<=n;i++) {
            int x = rd(), y = rd();
            b[x] ? b[y] = 1 :0;
            --a[x]; ++a[y];
            !a[x] ? a[x] = b[x] = 0 : 0;
        }

        for (int i=1;i<=3;i++) puts(b[i] ? "Yes" : "No");
    }

    return 0;
}
```

计算题

题目大意

给定正整数n，对n进行k次开平方根操作，输出结果（保留3位小数）。

解题思路

在k比较大的时候程序会超时。 $\lim_{k\to+\infty} n^{\frac{1}{2^k}} = 1$ 在k较大的时候答案会迅速向1靠近，在k比较大的时候输出1.000即可。（阈值设置成16时即可通过此题，即k大于16输出1.000） 使用其他方法，减少k较大时的计算，如二分答案再乘回去在数字比较大的时候退出，或者使用pow函数先计算指数都可以通过此题。

标程

```c
#include <stdio.h>
#include <math.h>
inline int rd() {int r;scanf("%d",&r);return r;}
inline int min(int a,int b) {return a<b ? a : b;}
int main() {
    int n = rd(), k = min(rd(), 16);
    double ans = n;
    for (int _=1;_<=k;_++) ans = sqrt(ans);
    printf("%.3f\n", ans);
    return 0;
}
```

Data structure alignment

题目大意

给定以下c/c++结构体内存对齐的规则，要求根据一给定结构体的成员定义顺序计算最终结构体所占空间字节数。

1. **对于结构体的各个成员，第一个成员的偏移量是0，排列在后面的成员其当前偏移量必须是当前成员类型的整数倍**
2. **结构体内所有数据成员各自内存对齐后，结构体本身还要进行一次内存对齐，保证整个结构体占用内存大小是结构体内最大数据成员的最小整数倍**
3. **如程序中有 `#pragma pack(n)` 预编译指令，则规则1中每个数据成员的对齐按照 `#pragma pack` 指定的对齐参数n和这个数据成员自身长度中比较小的那个进行；规则2中不再考虑最大结构体内类型，直接以该对齐参数进行对齐，且参数n的取值只能为1，2，4，8，16**

解题思路

按照题意模拟即可。

数据点0为样例；数据点1为对齐参数为1的情况；数据点2为无对其参数的情况；数据点3为所有可能情况。

模拟时只要记录当前偏移字节数，按照需要整除的对齐参数/成员大小进行调整，最后以最大的成员所占字节或给定的对齐参数对结构体进行最终的大小调整。

标程

```c
#include <stdio.h>
#include <string.h>
#include <assert.h>
int a[10001];
```

```
int main()
{
  int T;

  scanf("%d", &T);
  assert(T >= 1 && T <= 100);
  while (T--)
  {
    int N, Len, i;
    int ans = 0;
    scanf("%d", &N);
    scanf("%d", &Len);
    assert(N >= 1 && N <= 100);
    assert(Len == -1 || Len == 1 || Len == 2 || Len == 4 || Len == 8 || Len == 16);
    for (int i = 1; i <= N; i++)
    {
      char str[10];
      scanf("%s", str);
      if (strcmp("BYTE", str) == 0)
        a[i] = 1;
      else if (strcmp("WORD", str) == 0)
        a[i] = 2;
      else if (strcmp("DWORD", str) == 0)
        a[i] = 4;
      else if (strcmp("QWORD", str) == 0)
        a[i] = 8;
      else
      {
        printf("%s\n", str);
        assert(0);
      }
    }
    if (Len == -1)
    {
      for (int i = 1; i <= N; i++)
        Len = Len < a[i] ? a[i] : Len;
    }
    for (int i = 1; i <= N; i++)
    {
      int l = Len < a[i] ? Len : a[i];
      if (ans % l != 0)
        ans = ans + (l - (ans % l));
      ans += a[i];
    }
    if (ans % Len != 0)
      ans = ans + (Len - (ans % Len));
    printf("%d\n", ans);
  }
}
```

再次感谢大家的参与！