第13届趣味C决赛题解

> 对题目难度的估计过于乐观了呜呜。。。希望没有太影响到大家的心情

Solution for 救救孩子

题目大意

给定一个浮点数，是一个整数除以7的结果，保留两位小数，浮点数最后一位算错了，将它更正过来。

解题思路

答案和整数部分无关，只需要考虑小数部分。

形如$\frac{x}{7}(0 \leq x \leq 6)$的形式，总共只有七种情况，判断一下它属于哪一个即可。

参考数据

0/7 = 0.000000， 1/7 = 0.142857， 2/7 = 0.285714， 3/7 = 0.428571

4/7 = 0.571428，5/7 = 0.714285， 6/7 = 0.857142 。

这些数小数点后第一位都是不一样，根据数据修改最后一位即可，注意四舍五入。

庭庭的布偶

题目大意

给定三个长度相等的字符串。指定其中一个串，分别询问它是否能够排在第一/二/三的位置（可自定义字典序）。

解题思路

考虑分类讨论。

首先找到三个字符串第一个出现分歧的位置，此时会有两种情况：1、三个字母各不相同 2、其中一个字母不同于另外两个。

对于第一种情况，毋庸置疑输出三个"YES"，因为我们可以任意地调整三者的优先性。

对于第二种情况，不妨假设这三个字母分别为'A'、'B'、'B'。如果第K个布偶该位置正好是A，显然它没有办法排在中间，但一定可以排在两边，此时输出"YES"、"NO"、"YES"；如果第K个布偶该位置是B，我们便需要对这两个B打头的后缀子串继续遍历，找到它们第一次分歧的位置，此时，若第K个串出现一个'B'而另一个串出现了'A'，则说明庭庭偏爱的布偶无法排在中间，若第K个串出现一个'A'而另一个串恰好出现了'B'，则说明庭庭偏爱的布偶无法排在两边，剩下的情况均不会出现任何矛盾，讨论结束。

特别地，如果三个字符串完全相同，则说明任何排列方案都合法，直接输出三个"YES"。

标程

```
#include<stdio.h>
#include<string.h>
int K,len;
char str[5][110],ss[110];
int main(){
    scanf("%s%s%s",str[1],str[2],str[3]);
    len=strlen(str[1]);
    scanf("%d",&K);
    for(int i=0;i<len;i++){
        if(str[1][i]==str[2][i] && str[2][i]==str[3][i])continue;
        if(str[1][i]!=str[2][i] && str[1][i]!=str[3][i] && str[2][i]!=str[3][i]){
            printf("YES\n");printf("YES\n");printf("YES\n");return 0;
        }
        int tot=0;
        for(int ii=1;ii<=3;ii++)
            if(str[ii][i]!=str[K][i])tot++;
        if(tot==2){
```

```
            printf("YES\n");printf("NO\n");printf("YES\n");return 0;
        }
        char ch1=str[K][i],ch2;
        int flag1=1,flag2=1,a;
        for(a=1;a<=3;a++)if(str[a][i]!=ch1)ch2=str[a][i];
        for(a=1;a<=3;a++)if(a!=K && str[a][i]==str[K][i]){
            for(int ii=i+1;ii<len;ii++){
                if(str[a][ii]==str[K][ii])continue;
                if(str[a][ii]==ch1 && str[K][ii]==ch2)flag1=0;
                if(str[a][ii]==ch2 && str[K][ii]==ch1)flag2=0;
                break;
            }
            break;
        }
        if(flag1==1)printf("YES\n");else printf("NO\n");
        if(flag2==1)printf("YES\n");else printf("NO\n");
        if(flag1==1)printf("YES\n");else printf("NO\n");
        return 0;
    }
    printf("YES\n");printf("YES\n");printf("YES\n");
    return 0;
}
```

Solution for FA♂

一点屁话

这是一个解码题（比去年的简单吧ovo

题目大意

将读入的字符串（仅包含大写字母和空格），做以下操作

- 空格不变
- 设大写字母为ch，将 (ch - 'A') 表示为二进制形式，二进制码中 1用 F 代替，0 用 A 代替

输出转换后的字符串

得到题意的方法

很多；

观察样例可能没有什么效果；

可以观察到后面两个副标题中的第一个单词都是FAAFAAAAAAAFFAAAFFFFAFAFFAAFAA，鉴于这个标题对应的是输入输出样例，可以猜到这两个标题的含义是 Sample Input 和 Sample Output， 遂舒服；

还可以关注到每两个空格之间的单词长度都是5的倍数，26个字母，$2^5 = 32$

。。。

加密前的题意

Firstly, we denote alphabet A to Z with zero to twenty-five respectively. Then each code can be represented in binary form with five bits. Finally we transform one to F and zero to A, get a FA string. Given a string with only uppercase letters and spaces, transform it to a FA string.

Input

One line contains a string with only uppercase letters and spaces. The length is no longer than one hundred.

Output

One line contains the generated FA string.

Sample Input

A A

Sample Output

AAAAA AAAAA

Variadic Function 题解

题目本身没什么问题，就是写个简单的 `printf`，照着题目给的例子写就是了，不过有一个坑就是 C 语言参数传递时的自动类型提升。

`char` 会被 cast 成 `int` 再传过去，所以在拿的时候也得拿一个 `int` 。

这个信息可以在 pintia 给你的编译输出里看到的，所以不要随便忽略警告。

标程：

```c
int printNum(int n) {
    return printf("%d", n);
}

int printFloat(double f) {
    return printf("%lf", f);
}

int printStr(char* str) {
    return printf("%s", str);
}

int simple_printf(const char* fmt, ...) {
    va_list args;
    va_start(args, fmt);
    int counter = 0;
    while (*fmt != '\0') {
        if (*fmt != '$') { putchar(*fmt); fmt++; counter++; continue; }
        fmt++;
        if (*fmt == 'd') { // integer
            int int_val = va_arg(args, int);
            counter += printNum(int_val);
            fmt++;
        } else if (*fmt == 'f') { // double
            double float_val = va_arg(args, double);
            counter += printFloat(float_val);
            fmt++;
        } else if (*fmt == 's') { // char*
            char* str = va_arg(args, char*);
            counter += printStr(str);
            fmt++;
        } else if (*fmt == 'c') { // char
            char cha_val = va_arg(args, int);
            counter += 1;
            putchar(cha_val);
```

```
            fmt++;
        } else if(*fmt == '$') {
            fmt++;
            putchar('$');
            counter += 1;
        }
    }
    va_end(args);
    return counter;
}
```

Macro Overload 题解

这题做出来需要一些技巧。

1. 如何拿部分分？ 由于函数题只能设置一种检查程序，所以只有编译通过才能获得部分分。

很简单，直接把多于一个的参数都扔了就能过参数个数为 1 的那个点了，于是 20 分到手

```
#define MAX(X, ...) X
```

1. 正解怎么写？

需要一丁点技巧，不过也没什么好说的，看了标程就懂了。大概就是利用参数个数不同使固定位置上获取的参数不同从而实现基于参数个数的宏函数重载

标程：

```
#define MAX1(A) (A)
#define MAX2(A,B) (A)>(B)?(A):(B)
#define MAX3(A,B,C) MAX2(MAX2(A,B),(C))
#define MAX4(A,B,C,D) MAX2(MAX3(A,B,C),(D))
#define MAX5(A,B,C,D,E) MAX2(MAX4(A,B,C,D),(E))
#define GETMAX(A,B,C,D,E,MAXNAME,...) MAXNAME
#define MAX(...) GETMAX(__VA_ARGS__,MAX5,MAX4,MAX3,MAX2,MAX1)(__VA_ARGS__)
```

当然也不止这一种写法，可以随意发挥 (……) 。

Garbage Collection 题解

题面很长，抄了很多东西，但题目很简单，因为并没有让你收垃圾，而且数据量很小。

只需要从每个活跃的根去标记一下可达的内存数量和大小就可以了。

```c
#include <stdio.h>
#include <assert.h>

#define MAXN 10000
#define MAX_ID 10000
#define MAX_OBJ 10000

typedef struct _Node {
    size_t count;
    int vis;
    struct _Node** array;
} Node;
Node* create_node(int n) {
    Node* result = (Node*)malloc(sizeof(Node));
    result->count = n;
    result->vis = 0;
    result->array = (Node**)malloc(n * sizeof(Node*));
    for (int i = 0; i < n; i++) result->array[i] = NULL;
    return result;
}

Node* root[MAXN];
int mapping[MAX_ID];

Node* objects[MAX_OBJ];

void new_object(int object_id, int size) {
    objects[object_id] = create_node(size);
}

void new_gc_root(int root_id) {
    // do nothing
}

void remove_root(int root_id) {
```

```
    root[root_id] = NULL;
}

void link_to_object(int from, int offset, int to) {
    objects[from]->array[offset] = objects[to];
}

void link_to_root(int root_id, int object_id) {
    root[root_id] = objects[object_id];
}

int vis[MAX_OBJ];
Node* stack[MAX_OBJ];
int top;

int main(int argc, char** argv) {
    // freopen(argv[1], "r", stdin);
    int n;
    scanf("%d", &n);
    char oper[100];
    char type[100];
    int id;
    int size;
    int sum = 0;
    int reach = 0;
    for (int t = 0; t < n; t++) {
        scanf("%s", oper);
        // new
        if (oper[0] == 'n') {
            scanf("%s", type);
            // create object
            if (type[0] == 'o') {
                scanf("%d %d", &id, &size);
                sum += size;
                new_object(id, size);
            } else {
                // new gc root
                scanf("%d", &id);
                new_gc_root(id);
            }
        } else if (oper[0] == 'l') {
            int k;
            scanf("%d %d %d", &id, &size, &k);
            link_to_object(id, size, k);
        } else if (oper[0] == 'a') {
            scanf("%d %d", &id, &size);
```

```
            link_to_root(id, size);
        } else if (oper[0] == 'r') {
            scanf("%d", &id);
            remove_root(id);
        } else {
            break;
        }
    }
    for (int i = 0; i < MAX_ID; i++) {
        if (root[i]) {
            stack[top++] = root[i];
        }
    }
    int cnt2 = 0;
    while (top) {
        Node* cur = stack[--top];
        if (cur->vis) continue;
        cur->vis = 1;
        reach += cur->count;
        cnt2 ++;
        for (int i = 0; i < cur->count; i++) {
            if (cur->array[i] && !cur->array[i]->vis) {
                stack[top++] = cur->array[i];
            }
        }
    }
    printf("%d %d\n", reach, cnt2);
}
```

Solution for Math Strikes Back

题目大意

若正整数a符合在6进制下不存在某个数码(012345)既在a中出现、又在a^2中出现，则称a为好数。

解题思路

可以发现确定a最小的k位，就能确定a^2最小的k位。

从低到高枚举每一位，过程中如果不符合条件就return即可。

注意某些弱智（出题人）要跑满30位的所有情况。

最后把表打出来即可。

参考数据

第500个是225255525525525522555555555255。

祝各位在未来能有收获更多有趣的编程经历~