C8-2021级航类-第八次上机题解

A 选 择 题

难度	考点
10	综合应用

答案

CADCC

选项解析

第一题

- A：判等==的优先级高于位运算&，所以先进行11==10，其为假，也就是值为0，所以这条if语句等价于 if(x & 0)，无论x是多少，结果都是0
- B：小写字母都在大写字母后面
- D：unsigned int的最大值是4294967295，int的最大值是2147483647，并不是2倍的关系

第二题

函数与递归，直接模拟递归的过程就好了，题目要求计算 try(5)

> try(5) = 5 * try(3)
>
> - try(3) = 3 * try(1)
> - - try(1) = 1 * try(-1)
>
> - - - try(-1) = 1
> - - try(1) = 1 * 1 = 1
>
> - try(3) = 3 * 1 = 3
>
> try(5) = 5 * 3 = 15

希望大家明白递归一层一层往下然后回溯的过程

第三题

选D，p是个野指针，并没有指向某一个内存地址，不能对其进行*解引用

第四题

C选项，计算的是指针变量所占的字节大小（一般为8字节），而不是数组a所占的字节大小，故C选项错误
容易有疑问的可能是D选项，对int型指针p解引用之后，是一个int值，所以sizeof(*p)就是一个int所占的字节大小，再乘以N就是数组a所占的字节大小

第五题

选C.argv[0]是程序路径名,argv[2]就是第二个输入的参数.++argv[2]就从第2个参数的第2个字符开始输出字符串,

B 计 算 器

难度	考点
1	分支结构

题目解析

本题为入门题，仅需按照题目要求写出 `if` 分支结构即可，注意除法运算需要强制类型转换即可。

示例代码

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>

int main()
{
    int a,b;
    char c;
    while(scanf("%d %c %d",&a,&c,&b)!=EOF)
    {
        if(c=='+')
        {
            printf("%d\n",a+b);
        }
        if(c=='-')
        {
            printf("%d\n",a-b);
        }
        if(c=='*')
        {
            printf("%d\n",a*b);
        }
        if(c=='/')
        {
            printf("%.4lf\n",(double)a/b);
        }
        if(c=='%')
        {
            printf("%d\n",a%b);
        }
    }
    return 0;
}
```

C 鸡尾酒疗法

难度	考点
1	数据类型

题目分析

计算鸡尾酒疗法的有效率公式是：

鸡尾酒疗法的有效率=“疗效有效的病例数”÷“临床实验的总病例数”

所以，根据此公式，我们可以先计算第一组鸡尾酒疗法的有效率，与其他 n−1 组的改进疗法的有效率进行比较，计算差值与 5% 的关系，根据题目要求输出对应的答案即可。

示例代码

```
#include <stdio.h>
int n,a,b;
int main() {
    scanf("%d%d%d",&n,&a,&b);
    int c, d;
    double x=1.0*b/a,y;
    for(int i=0;i<n-1;i++)
    {
        scanf("%d%d",&c,&d);
        y=1.0*d/c;
        if(y-x>=0.05) puts("better");
        else if(x-y>=0.05) puts("worse");
        else puts("same");
    }
    return 0;
}
```

D 取反数

难度	考点
2	位运算

题目分析

（做法有很多，以下给出一种解答方法）读完题目首先想到使用 `~` 按位取反，但是直接令 `num = ~num` 会把 num 的二进制高位 0 都变为 1，只需要再把高位都清零即可。

示例代码

```c
#include <stdio.h>

int main() {

    int num, i;
    scanf("%d", &num);
    num = ~num;

    for (i = 31; i >= 0; i--) {
        if (num >> i & 1) {          // 取出二进制第i位
            num &= ~(1 << i);        // 第i位为1，将其清零
        } else {                     // 第i位为0，则高位都已清零，跳出循环
            break;
        }
    }

    printf("%d\n", num);

    return 0;
}
```

E 卡牌数数

难度	考点
2	一维数组

问题分析

题目最主要的问题在于将一个数x的每一位提取出来并统计个数：

假设$x = 1234$，我们可以通过**%**运算获得x的最后一位4，统计最后一位后我们直接将x整除10，此时x变为123，我们再通过**%**运算获得x的最后一位3，后面依次循环往复，即可提取出x的每一位数。

之前的练习中有过类似的题目，相信大家能很快的想到。

参考代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define ll long long
int cnt[10];
int check(ll x) //判断x是否能被余下的卡牌表示出来
{
    int now[10]={0}; //用来存储表示x所需的每种卡牌的数量
    while(x) //拆分出x的每一位
    {
        now[x%10]++; //x%10提取出当前最低位的数
        x/=10; //将最低位的数舍去
    }
    for(int i=0;i<10;i++)
    {
        if(now[i]>cnt[i]) return 0; //判断卡牌i是否够用，不够的话返回0
        cnt[i]-=now[i]; //更新卡牌数量
    }
    return 1; //所有卡牌均够用，返回1
}
int main()
{
    ll x;
    for(int i=0;i<10;i++) scanf("%d",&cnt[i]); //输入各张卡牌的数量
    scanf("%lld",&x); //输入初始数数的位置x
    for(;check(x);x++); //判断x是否可以被表示出来，如果可以则令x++，否则退出循环
    printf("%lld\n",x-1); //退出循环时的x是无法被表示的，所以最多数到x-1
    //system("pause");
    return 0;
}
```

F 上帝的名单

难度	考点
3	结构体qsort

问题分析

由于题目要求$1 \leq n \leq 100000$数据量较大，所以容易想到需要对建立的结构体数组用qsort进行排序，`strcmp(str1,str2)`可以很好的比较两字符串的大小。

在练习赛中已经有过类似的结构体排序题目，希望大家能更熟练的掌握结构体的qsort写法。

参考代码

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAX 100050

typedef struct list
{
    char name[25];
    int num;
}List;

int cmp(const void*a,const void*b)
{
    int l1=((struct list*)a)->num;
    int l2=((struct list*)b)->num;
    if(l1!=l2)return l1 > l2 ? -1:1;
    else return strcmp(((struct list*)a)->name,((struct list*)b)->name);
}
List a[MAX];
int main(){
    int i,n=0;
    while(scanf("%s%d",a[n].name,&a[n].num)!=EOF)
    {
        if(a[n].num>=100)n++;
    }
    qsort(a,n,sizeof(a[0]),cmp);
    printf("%d\n",n);
    for(i=0;i<n;i++)
    {
        printf("%s %d\n",a[i].name,a[i].num);
    }
}
```

G 黑子的篮球训练

难度	考点
2	斐波那契数列

题目分析

黑子恰好得到 n 分，考虑他最后一次出手的选择：可以投两分球，那么此前得分为 $(n-2)$ 分，或是投三分球，则此前得分为 $(n-3)$ 分。因此假设 $ans(i)$ 表示得到 i 分的不同投球顺序总数，那么 $ans(i)=ans(i-2)+ans(i-3)$，可以看出实际上是斐波那契数列的变形，通过递推的方式求解数列的第 n 项。

示例代码

```c
#include <stdio.h>

int main() {

    // ans[i]表示得到i分的不同投球顺序总数，那么有ans[2]=ans[3]=1，先赋上初值
    int n, ans[100] = {0, 0, 1, 1, }, i;
    scanf("%d", &n);

    for (i = 4; i < n; i++)
        ans[i] = ans[i - 2] + ans[i - 3];       // 递推公式
    printf("%d\n", ans[n]);

    return 0;
}
```

H 呱呱泡蛙做毕业设计

难度	考点
2	字符串

题目分析

本题主要考察字符串库函数的应用。

示例代码

```
#include<stdio.h>
#include<string.h>

char input[55];
char instr[55];

char op[55];

void handler()
{
    if(strcmp(instr,"strlen")==0)
    {
        printf("%d\n",strlen(input));
    }
    else if(strcmp(instr,"strstr")==0)
    {
        scanf("%s",op);
        char *loc=strstr(input,op);
        if(loc==NULL)
        {
            printf("NULL\n");
        }
        else
        {
            printf("%d\n",loc-input);
        }
    }
    else if(strcmp(instr,"strcpy")==0)
    {
        scanf("%s",op);
        strcpy(input,op);
    }
    else if(strcmp(instr,"strcat")==0)
    {
        scanf("%s",op);
```

```
        strcat(input,op);
    }
}

int InstrCnt;

int main()
{
    scanf("%s",input);
    scanf("%d",&InstrCnt);
    while(InstrCnt--)
    {
        scanf("%s",instr);
        handler();
    }
    printf("%s\n",input);
}
```

I 溢流

难度	考点
4	二分查找

题目分析

首先，注意到题目假设，将网格高度从小到大排序后进行处理

从原理上，水从最低处开始累积，每到一个新的水平面高度，高度的变化速率就会不同(因为底面积增加了，相同体积对应的高度不同)

考虑到算法效率问题，我们采用建表+二分查找的方式来确定当前输入水量的底面积，当然，由于本题的数据范围存在一些有趣的特点，大家也可以考虑使用哈希+线性查找等方法进行优化，也能通过本题。

示例代码

```
#include<stdio.h>
#include<stdlib.h>
long long v[640005];//体积
int h[640005];//高度
int n,m;
int cmp(const void *a , const void *b)
{
    return *(int *)a - *(int *)b;    //为什么可以这么写？注意数据范围
}
//求下界，既在单调递增序列中查找>=index的数中最小的一个（即index或index的后继）
int lower_bound(long long index)//这里用int也行，懒得改了
{
    int low = 1;
    int high = m*n;
    int mid;
    while (low < high)
    {
        mid = low + (high - low) / 2;
        if (v[mid] >=index) high=mid;
        else low = mid + 1;
    }
    return low;
}
int main(){
    int i,j;
    long long total;
    int fo;
    double ans;
    scanf("%d%d",&m,&n);
    for(i=0;i<m*n;i++)
```

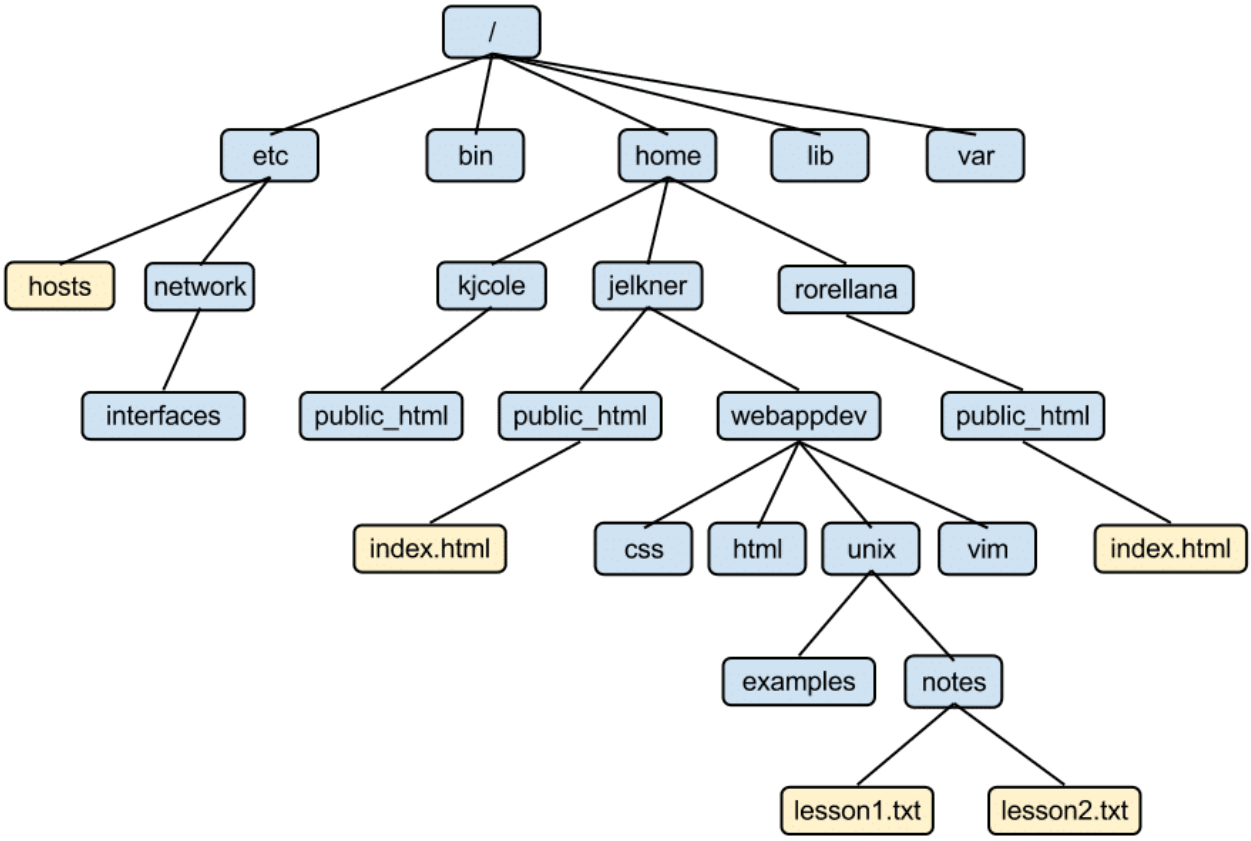
```
        scanf("%d",&h[i]);
    qsort(h,m*n,sizeof(h[0]),cmp);
    h[m*n] = 2147483647;
    for(i=1;i<=m*n;i++)
        v[i] = (long long)100*i*(h[i]-h[i-1])+v[i-1];

    while(scanf("%lld",&total)!=EOF)
    {
        fo = lower_bound(total);
        ans = h[fo-1]+(total - v[fo-1]) /100.0/fo;
        //高度=上一个高度+（总水量-上一个高度对应的水量）/当前底面积
        printf("%lf\n",ans);
    }
}
```

J sh-3.2# tree .

难度	考点
5	树、综合

题目解析

一个文件夹中的文件结构是一个典型的树形结构，可以形象地称为“文件树”。下图是一颗文件树。



因此要做这道题，需要知道采用什么样的数据结构来维护一棵树。树实际上是一种特殊的图，即无环图。形象一点的解释，除了最外层文件夹之外（图中的`/`文件夹），所有文件夹有且仅有一个父文件夹。

存储一颗树（图）通常使用到的方法是邻接表或邻接矩阵，在C语言中通常使用二维数组或链表来实现这种数据结构，邻接表/邻接矩阵的构建留给同学们自学。

本题输入数据的第一行t实际上是总文件夹数量减去一，即文件树中”边“的个数，接下来t行则描述了整颗文件树，给出了所有文件夹的父子关系，我们需要在读入这t行之后，用邻接表/邻接矩阵构建出这颗文件树。

接下来我们需要打印整颗文件树，为此我们需要首先找到最外层文件夹，也就是根结点。根据之前的解释，没有父文件夹的文件夹即是最外层文件夹。找到最外层文件夹后，如何按规定打印文件树呢？实际上树有”递归“的特性，即某个结点的子节点仍然是一棵树的根结点，因此我们可以用递归函数来处理，具体的思路之一参考实例代码。

最后是多组输入数据的处理，在能够打印出整棵树之后，实际上打印子文件夹的树就是简单的函数调用，唯一需要我们去做的就是在文件树中找到这个文件夹的位置。

示例代码1：二维数组+邻接表

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char items[1030][35]; // 用于存储文件夹名
int total_items = 0;
int find_item(char *s) {
    // 找到文件夹名为s的文件夹，返回其在items数组中的索引
    for (int i = 0; i < total_items; i++) {
        if (strcmp(s, items[i]) == 0) {
            return i;
        }
    }
    return -1;
}
int set_item(char *s) {
    // 如果文件夹名为s的文件夹不在items数组中，存储它。返回s在items数组中的索引
    int idx = find_item(s);
    if (idx == -1) {
        strcpy(items[total_items], s);
        idx = total_items++;
    }
    return idx;
}
char *get_item(int idx) {
    return items[idx];
}

int tree[1030][1030], node_cnt[1030]; // tree为邻接表，node_cnt维护每个结点的孩子数

int cmp(const void *pa, const void *pb) {
    int a = *(int *)pa, b = *(int *)pb;
    return strcmp(get_item(a), get_item(b));
}

void print_tree(int idx, char *prefix) {
    // 打印以items[idx]为根结点的树，prefix维护需要在文件夹名之前打印的字符串
    int len = (int)strlen(prefix);
    for (int i = 0; i < node_cnt[idx]; i++) {
        printf(i == node_cnt[idx] - 1 ? "%s`-- %s\n" : "%s|-- %s\n", prefix,
get_item(tree[idx][i])); //打印整行
        print_tree(tree[idx][i], strcat(prefix, i == node_cnt[idx] - 1 ? "    " : "|
")); // 递归调用
        prefix[len] = 0; // 还原prefix字符串
```

```
    }
}

int main(int argc, const char * argv[]) {

    char father[35], child[35];
    int have_father[1030] = {0}, t; // have_father数组维护结点是否有父亲
    scanf("%d", &t);
    while (t--) {
        scanf("%s %s", father, child);
        int fa_idx = set_item(father),  ch_idx = set_item(child);
        have_father[ch_idx] = 1;
        tree[fa_idx][node_cnt[fa_idx]++] = ch_idx; // 在邻接表中添加元素
    }

    int root = 0;
    for (int i = 0; i < total_items; i++) {
        if (have_father[i] == 0) {
            root = i;
        }
        qsort(tree[i], node_cnt[i], sizeof(int), cmp); //将结点的孩子按文件夹名字典序升序排序
    }

    char prefix[5000] = {0}, dir[35]; // 令prefix为空字符串
    printf("%s\n", get_item(root));
    print_tree(root, prefix); // 打印整棵树
    while (~scanf("%s", dir)) {
        printf("%s\n", dir);
        print_tree(find_item(dir), prefix); // 打印子树
    }

    return 0;
}
```

示例代码2：链表+邻接表

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct node {
    char name[35]; // 文件夹名
    int have_father; // 是否有父节点
    int child_num; // 孩子数量
    struct node *child[1024]; // 存储指向孩子结点的指针
};
struct node pool[1024]; // 预分配结构体空间
int total_nodes = 0;
```

```c
struct node *get_struct(void) { // 获取一个空的结构题
    return &pool[total_nodes++];
}

struct node *find_node_in_pool(char *name) {
    // 在pool数组中寻找文件夹名为name的结构体
    struct node *find = NULL;
    for (int i = 0; i < total_nodes; i++) {
        if (strcmp(pool[i].name, name) == 0) {
            find = &pool[i];
        }
    }
    if (find == NULL) {
        find = get_struct();
        strcpy(find->name, name);
    }
    return find;
}

int cmp(const void *pa, const void *pb) {
    struct node *a = *(struct node **)pa, *b = *(struct node **)pb;
    return strcmp(a->name, b->name);
}

struct node *sort_and_find_root(void) {
    // 将所有结点的孩子结点进行排序，并找到整棵树的根结点
    struct node *root = NULL;
    for (int i = 0; i < total_nodes; i++) {
        if (pool[i].have_father == 0) {
            root = &pool[i];
        }
        qsort(pool[i].child, pool[i].child_num, sizeof(struct node *), cmp);
    }
    return root;
}

void print_tree(struct node *tree, char *prefix) {
    // 打印以tree为根结点的树，prefix维护需要在文件夹名之前打印的字符串
    int len = (int)strlen(prefix);
    for (int i = 0; i < tree->child_num; i++) {
        printf(i == tree->child_num - 1 ? "%s`-- %s\n" : "%s|-- %s\n", prefix, tree-
>child[i]->name); //打印整行
        print_tree(tree->child[i], strcat(prefix, i == tree->child_num - 1 ? "    " :
"|   ")); //递归调用
        prefix[len] = 0; // 还原prefix字符串
    }
}

int main(int argc, const char * argv[]) {
```

```
    char father[35], child[35];
    int t;
    scanf("%d", &t);
    while (t--) {
        scanf("%s %s", father, child);
        struct node *fa_node = find_node_in_pool(father);
        struct node *ch_node = find_node_in_pool(child);
        ch_node->have_father = 1;
        fa_node->child[fa_node->child_num++] = ch_node; // 添加孩子结点
    }

    struct node *root = sort_and_find_root();

    char prefix[5000] = {0}, dir[35]; // 令prefix为空字符串
    printf("%s\n", root->name);
    print_tree(root, prefix); // 打印整棵树
    while (~scanf("%s", dir)) {
        printf("%s\n", dir);
        print_tree(find_node_in_pool(dir), prefix); // 打印子树
    }

    return 0;
}
```

K GCD求和

难度	考点
6	循环、最大公约数

问题分析

先考虑一个时间复杂度$O(N^2)$的做法：从左到右遍历a序列，设当前遍历到a_t，用一个数组b，b_i表示$gcd(a_i,a_{i+1},\dots,a_t)$的最大公约数。当要遍历到$a_{t+1}$时，遍历$b$数组，更新即可。答案就是$\sum_{t=1}^{n}\sum_{i=1}^{t}b_i$

如何优化时间复杂度？可以发现gcd存在一个性质，如果$gcd(x,y)\neq x$，那么$gcd(x,y)\leq\frac{x}{2}$，因此对于一个10^{18}数量级的数，不停和其他数取gcd，直到1的过程中，最多仅会出现$log_2 10^{18}$个不同的数。

回到题目，b数组也可以看作是a_t不停取gcd的结果，也就是说，b数组中最多仅仅会存在60个不同的数。把相同的数合并，这样我们可以把b数组压缩到长度只有64的小数组，记录下每个数的个数，同样可以统计答案。

时间复杂度：$O(NlogM)$，M为序列中最大的数。

参考代码

```
#include<stdio.h>
#include<stdlib.h>
#define mod 998244353

long long a[1000005],ans;
int n,cnt;

struct steins{
  int pos;
  long long x;
};

struct steins k[105];
long long gcd(long long x,long long y){
  if(y==0) return x;
  return gcd(y,x%y);
}

void merge(){
  int g=0;
  for(int i=1;i<=cnt;i++)
    if(k[i].x!=k[i-1].x)
      k[++g]=k[i];
  cnt=g;
}

int main()
```

```
{
  scanf("%d",&n);
  for(int i=1;i<=n;i++) scanf("%lld",&a[i]);
  for(int i=1;i<=n;i++){
    for(int j=1;j<=cnt;j++)
      k[j].x=gcd(k[j].x,a[i]);
    k[++cnt].x=a[i];
    k[cnt].pos=i;
    merge();
    k[cnt+1].pos=i+1;
    for(int j=1;j<=cnt;j++)
      ans=(k[j].x%mod*(k[j+1].pos-k[j].pos)+ans)%mod;
  }
  printf("%lld",ans);
}
```