逻辑题

1. 参加断型冠状病毒疫苗开发研讨会的70名学者中,亚商学者39人,博士33人,非亚裔学者中无博士学位的4人，根据以上陈述,参加此次研讨会的亚裔博士有几人?

A 1

B 2

C 6

D 7

E 8

> C
>
> 亚裔学者39人，博士33人，非亚裔学者中无博士学位4人，这三者加起来是76人，但实际总人数只有70人。亚裔学者和博士两个概念之间为交叉关系，这两个概念和非亚裔学者中无博士学位者之间都是全异关系。这说明，既是亚裔学者又是博士即亚裔博士有6人。
>
> 也可以通过运用计算法来求解。设亚裔博士有x人，则可列方程：31+33-x+4=70，解这个方程，可得：x=6。

2. 某省妇女儿童占全省总人口的2/3。如果妇女是指所有女性人口，儿童是指所有非成年人口，并且对任一年龄段，该省男女人口的数量持平，则上述断定能推出以下哪项结论？

A 该省男性成年人口和儿童人口持平。

B 该省男性成年人口大于儿童人口。

C 该省男性成年人口小于儿童人口。

D 该省女性成年人口和男性儿童人口持平

E 该省男性成年人口和女性儿童人口持平。

> A
>
> 由题干，可以给出以下表
>
> | | **女性** | **男性** | |
> |---|---|---|---|
> | 成年 | 1/3 | 1/3 | 2/3 |
> | 儿童 | 1/6 | 1/6 | 1/3 |
> | | 1/2 | 1/2 | |
>
> 由任一年龄段，该省男女人口的数量持平，可得总人口男女持平。
>
> 由妇女儿童占全省总人口的三分之二，可得成年男性占三分之一。

> 由成年男性占三分之一，得男童占六分之一（因为男性占二分之一）。
>
> 由男童占六分之一，得女童占六分之一。因此，该省男性成年人口和儿童人口持平。

3. 某次讨论会共有25名与会者,已知(1)至少有7名青年教师是男性，(2)至少有8名男教师已过中年(3)至少有10名男青年是教师;如上述3句话两真一假,则关于与会人员可以得出以下哪项

A 青年教师至少有10名

B 男教师至多有15名

C 男青年都是教师

D 男青年至少有7名

> D
>
> [1]：大于等于7名青男 [2]：大于等于8名中男[3]：大于等于10名青男。
>
> [1]与[3]数量上有重叠部分，如果[3]为真，则[1]一定为真；
>
> 如果[1]为假，则[3]一定为假。此时就会两个为假，与题干条件的两真一假相矛盾，故[1]必真。而如果[1]为真，既青年男教师大于等于7人，那么男青年至少有7名。

4. 某中药配方有如下要求(1)如果有甲药材,那么没有乙药材(2)如果没有丙药材,那么必须有丁药材(3)人参和天麻不能都有(4)如果没有甲药材而有丙药材,则需要有人参。如果还有天麻,则关于该配方的断定哪项为真?D

A 含有甲药材

B 含有丙药材

C 没有丙药材

D 含有乙药材或不含丁药材

> 无正确答案
>
> 由“含有天麻”和（３）可以推出，不含有人参；进而由（４），否定后件就能否定前件，可以推出，有甲药材或者没有丙药材。
>
> 如果有甲药材，由（１）可以推出，**无乙药材**；如果没有丙药材，由（２）可以推出，**有丁药材**；故**无乙药材或有丁药材**。
>
> 因此，选项中没有正确答案。

5. 某国拟在甲、乙、丙、丁、戊己6种农作物中进口几种,用于该国庞大的动物饲料产业,考虑到些农作物可能有违禁成分,以及它们之间存在的互补或可替代因素,该国对进口这些农作物有如下要求(1)它们当中不含违禁成分的都进口。2)如果甲或乙含有违禁成分,就进口丙和丁。3)如果戊含有违禁成分,那么己就不进口了;如果进口丙,就进口乙和己。(4)如果不进口己,就进口戊;如果进口戊,就不进口己。根据上述要求,以下哪项所列的农作物是该国可以进口的；

A 甲、乙、丙

B 乙、丙、丁

C 甲、乙、戊

D 甲、乙、己

E 丙、戊、己

> C
>
> A选项与（2）矛盾
>
> B选项与（2）矛盾
>
> D选项与（3）矛盾
>
> E选项与（4）矛盾

不定向选择

1. int i =1;const int j =2;以下说法不正确的是

A const int *p1 = &i;

B const int *p2 = &j;

C int *const p3 = &i;

D int *const p4 = &j;

> D
>
> int *const p4 ,p4为指针常量，p4指向的内存位置不能改变，但是，p4所指内存存放的值是可以改变的。j表示常量，其数值不能被改变。
>
> 将j的地址赋给p4后，**p4可以执行其他操作**（如*p4=4;），将j的值改变，因此，int *const p4 = &j;是错误的。

2. 以下关于内存的说法正确的是

A RAM是随机存储器，在断电时将丢失其存储内容，ROM是只读存储器，断电时不会丢失存储内容

B 内存的数据带宽与内存的数据传输频率、内存数据总线位数以及内存大小有关

C 用户进程通常情况只能访问用户空间的虚拟地址,不能访问内核空间虚拟地址

D Linux中使用 buddy system算法可以管理页外内存碎片,使用slub算法可以管理页内内存碎片

> ACD
>
> B:内存的数据带宽的计算公式是：数据带宽=内存的数据传输频率×内存数据总线位数/8

3. 以下哪些事件会导致进程的创建

A 系统初始化

B fork系统调用

C pthread_ create函数调用

D 一个批处理作业的初始化

> ABD
>
> 创建进程的多种方式但凡是硬件，都需要有操作系统去管理，只要有操作系统，就有进程的概念，就需要有创建进程的方式，一些操作系统只为一个应用程序设计，比如扫地机器人，一旦启动，所有的进程都已经存在。
>
> 而对于通用系统（跑很多应用程序），需要有系统运行过程中创建或撤销进程的能力，主要分为4中形式创建新的进程
>
> 1.系统初始化（查看进程 linux中用ps命令， windows中用任务管理器，前台进程负责与用户交互，后台运行的进程与用户无关，运行在后台并且只在需要时才唤醒的进程，称为守护进程，如电子邮件、web页面、新闻、打印）
>
> 2.一个进程在运行过程中开启了子进程（如 nginx开启多进程，os.fork等）
>
> 3.用户的交互式请求，而创建一个新进程（如用户用鼠标双击任意一款软件，qq，微信等）
>
> 4.—个批处理作业的初始化（只在大型机的批处理系统中应用）
>
> 无论哪一种，新进程的创建都是由—个已经存在的进程执行了—个用于创建进程的系统调用而创建的。

4. 下列说法正确的有

A 计算机体系结构是一门研究计算机系统软件结构的学科。

B 现代计算机处理器结构按照存储方式划分,可分为复杂指令集计算机和精简指令集计算机

C RISC技术对比CISC最大的区别就是对CPI的精简

D 单指令流单数据流计算机的每个机器周期最多执行一条指令

> CD
>
> A.计算机体系结构主要研究软件、硬件功能分配和对软件、硬件界面的确定
>
> B.现代计算机处理器结构按照**指令系统**方式划分,可分为复杂指令集计算机和精简指令集计算机

5. 32位系统中，该程序的输出为

```
//参数传递 退化为指针
void Func(char str_arg[100])
{
    printf("%d\n",sizeof(str_arg));
}
int main()
```

```
{
    char str[] = "Hello";
    printf("%d\n",sizeof(str));
    printf("%d\n",strlen(str));
    char *p = str;
    printf("%d\n",sizeof(p));
    Func(str);
    return 0;
}
```

A 5 5 4 4

B 6 5 4 4

C 6 5 6 4

D 5 5 5 100

> B 6 5 4 4
>
> 使用函数**strlen()**求某个字符串的长度时是**不包括结尾标志符'\0'**的，但当你用**sizeof()**求某个字符串占用的内存空间时，**结尾字符'\0'是被包括在里面的。**
>
> strlen用来计算字符串的长度（在C/C++中，字符串是**以"\0"作为结束符的**），它从内存的某个位置（可以是字符串开头，中间某个位置，甚至是某个不确定的内存区域）开始扫描直到碰到第一个字符串结束符\0为止，然后返回计数器值。
>
> sizeof是C语言的关键字，它以**字节的形式**给出了其操作数的**存储大小**，操作数可以是一个表达式或括在括号内的类型名，操作数的存储大小由操作数的类型决定。

6. 有以下程序,求输出结果

```
#include<stdio.h>
int fun(int i)
{
    int cnt = 0;
    while(i)
    {
        cnt++;
        i=i&(i-1);
    }
    return cnt;
}
int main()
{
    printf("%d\n\r",fun（2021));
    return 0;
}
```

> 8
>
> &是按位与，对应位都为1时该位得1，否则得0。所以 i&(i-1) 的作用：将i的二进制表示中的最右边的1置为0。
>
> 在本题中即数出2021转换成二进制有几个1就会走几次循环(不断除2)。2021对应的二进制是：10100111111，一共8个1，故走8次。
>
> 扩展：(n > 0 && ((n & (n - 1)) == 0)是判断n是不是2的次幂

7. 若 int x = 5&6,那么x的值为（ ）

A 3

B 4

C 5

D 6

```
B
5: 0101
6: 0110
x: 0100
```

8. 以下错误的表达式为

```
struct {
    inr a;
    char b;
}Q,*p=&Q;
```

A Q.a

B (*p).b

C p->a

D *p.b

> D
>
> *p=&Q ，把Q的地址赋值给了指针p，对p解引用其实就是Q。
>
> A 选项肯定是对的，结构体的正常访问方法。
>
> B 选项 (*p).b 等价于 Q.b
>
> C p->a p为指针访问结构体用->没问题。

D *p.b 优先级问题，.的优先级高于 *，所以 *p.b == *（p.b）， p为指针，访问结构体成员要用->。

扩展：结构体中.和->两种访问区别

定义结构体指针，访问成员时就用->

定义结构体变量，访问成员时就用.

```
struct A {
 int a;
 char b;
};
struct A q; //访问成员就用：q.a;
struct A *p; //访问成员就用：p->a;
```

9. 关于对象的this指针,以下叙述不正确的有

A 必须显示地在类中定义声明this数据成员才能使用this指针

B 一旦生成一个对象,该对象的this指针就指向该对象本身

C 一个类的所有对象的this指针的值都是相同的

D 不能通过对象的this指针访问对象的数据成员和成员函数

A

this指针的特点：

(1) 每个当前对象都含有一个指向该对象的this指针。this指针只能在类的成员函数中使用，在全局函数、静态成员函数中都不能使用 this 。

(2) this 指针是在成员函数的开始前构造，并在成员函数的结束后清除 。

(3) this 指针会因编译器不同而有不同的存储位置，可能是寄存器或全局变量 。

(4) this 是类的指针 。

(5) 因为 this 指针只有在成员函数中才有定义，所以获得一个对象后，不能通过对象使 用 this 指针，所以也就无法知道一个对象的 this 指针的位置。 不过，可以在成员函数中指定this 指针的位置 。

(6) 普通的类函数（不论是非静态成员函数，还是静态成员函数）都不会创建一个函数表来保存函数指针，只有虚函数才会被放到函数表中。

10. 若某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素,则最节省运算时间的存储方式是

A 单链表

B 仅有头指针的单循环链表

C 双链表

D 仅有尾指针的单循环链表

> D
>
> 单链表只能单向遍历，即只能由链表头向链表尾遍历。
>
> 单循环链表也只能单向遍历：链表头->链表尾->链表头；
>
> 对于A，B，C要想在尾端插入结点，需要遍历整个链表。
>
> 对于D，要插入结点，只要改变一下指针即可，要删除头结点，只要删除指针.next的元素即可。
>
> 如果只要知道尾指针p，则通过计算一次p->next就能获得头指针；插入和删除算法复杂度O(1)+O(1)
>
> 而如果只知道头指针，则需要遍历整个链表来获得尾指针的位置;插入和删除算法复杂度O(1)+O(N)
>
> 所以D仅有尾指针的单循环链表存储方式最节省运算时间

填空题

1. F和Q分别是指向单链表两个元素的指针,那么,F所指元素是Q所指元素后继的条件是（Q->next == F）

2.设有一个空栈,现有输入序列为1，2，3，4，5，经过 push, push, pop, push pop, push, pop, push后,输出序列是（2,3,4）

> 2,3,4
>
> push进栈，栈中是1
>
> push进栈，栈中是1，2
>
> pop出栈，栈中是1，**输出2**
>
> push进栈，栈中是1，3
>
> pop出栈，栈中是1，**输出3**
>
> push进栈，栈中是1，4
>
> pop出栈，栈中是1，**输出4**
>
> push进栈，栈中是1，5

3.H是一个带有头结点的单链表,那么在第一个元素之后插入一个结点（设P为要插入节点的指针），其操作是（p->next=H->next;）（H->next = p;）

4.已知二叉树后序遍历序列为dabec，中序遍历序列为debac，它的前序遍历序列为

> cedba

1、由后续遍历可知c是根结点。

2、由中序遍历可知deba在c的左孩子树上。

3、由后序遍历知e是c的左孩子树的根结点。

4、由中序遍历可知d是e的左孩子，ba在e的右孩子树上。

5、由后序遍历，可以得出b是e的右孩子，a是e的左孩子，而第4步中确定了a不在e的左孩子数上，因此，a只能在b上。

6、由中序遍历，可知，a是b的右孩子。

所以前序序列为cedba，具体如下图所示。

5.以下程序的输出结果为

```
#include <stdio.h>
int main()
{
    int a;
    a=(int)((double)(3/2)+0.5);
    printf("a = %d",a);
    return  0;
}
```

1

3和2是整形常量，所以3/2=1；前面（double） 1 = 1.000000；1.000000+0.5 = 1.500000；double转int会直接去掉小数部分。所以答案为1。

下面简单分析下double转int为什么会舍去小数部分

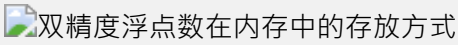
根据国际标准 IEEE 754，任意一个二进制浮点数 V 可以表示成下面的形式：

V = (-1) ^ s × M × 2 ^ E

（1）(-1)^s 表示符号位，当 s=0，V 为正数；当 s=1，V 为负数。

（2）M 表示有效数字，大于等于 1，小于 2，但整数部分的 1 不变，因此可以省略。M由frac编码。

（3）2^E 表示指数位。E由exp编码。

 双精度浮点数在内存中的存放方式

对于 64 位的双精度数来说，从低位到高位，尾数 M 用 52 位来表示，阶码用 11 位来表示，而符号位用最高位 1 位来表示，0 表示正，1 表示负。

将1.5转换为双精度浮点数的过程如下：

1. 将十进制数1.5转换成二进制为1.1。
2. 1.1用二进制的科学计数法表示为1.1 * 2^0
3. 按照上面浮点数的存储结构， 得出符号位为： 0，表示正数；阶码（指数） E 为1023； 小数部分 M 为1。
4. 双精度的二进制位：
   0_01111111111_0000000000000000000000000000000000000000000000000001

int类型为32位，double转换为int只能截取低32位为00000000000000000000000000000001。

所以最终的输出结果为1。

6.这段程序会存在什么问题

```
#include <stdio.h>
void my_alloc(char **p)
 {
    *p = (char *)malloc(100);
}
intm main()
{
    char *otr = NULL;
    char *ptr = NULL;
    my_alloc(&ptr);
    strcpy(ptr,"hello world");
    printf("%s\n",ptr);
    return 0;
}
```

内存泄漏，分配给其他变量的内存就会减小。**我们在删除一个指针之后，编译器只会释放该指针所指向的内存空间，而不会删除这个指针本身。此时p也就成为一个野指针**

扩展:

free()到底释放了什么？

free()释放的是**指针指向的内存！注意,释放的是内存，不是指针。**

指针是一个变量，只有程序结束时才被销毁。释放了内存空间后，原来指向这块空间的指针还是存在,只不过现在指针指向的内容的垃圾，是未定义的，所以说是垃圾。

因此，释放内存后把指针指向NULL，防止指针在后面不小心又被解引用了。当然，具体情况要具体分析以及具体解决。比如说，你定义了一个指针，在一个函数里申请了一块内存，然后通过函数返回传递给这个指针，那么也许释放这块内存这项工作就应该留给其他函数了。

malloc()到底从哪里得到了内存空间？

从堆里面获得空间。也就是说函数返回的指针是指向堆里面的一块内存。操作系统中有一个记录空闲内存地址的链表。当操作系统收到程序的申请时，就会遍历该链表，然后就寻找第一个空间大于所申请空间的堆结点，然后就将该结点从空闲结点链表中删除，并将该结点的空间分配给程序。

7.改正程序中的错误

下列给定程序将数组元素循环右移，数组大小和元素以及移动位数由键盘输入指定，例如数组{ 1,2,3,4,5,6}，循环右移三位，得到结果为{ 4,5,6,1,2,3}

请改正程序中的错误，使它能得出正确的结果。注意：不能更改程序的结构（共有四处错误）

```
#include<stdio.h>
#include<stdlib.h>
void shift_func(int *array, int len, int k)
{
    int i = 0, j = 0;
    int temp = 0;
    if (array == NULL)
        return;
    k %= len;
    for (i = 0; i < k; i++)
    {
        temp = array[len - 1];
        for (j = len; j > 0; j--)
        {
            array[j] = array[j - 1];
        }
        array[0] = temp;
    }
}
int main()
{
    int len = 0, k = 0, i = 0;
    int *array = NULL;

    scanf("%d%d", &len, &k);
    array = (int*)malloc(len * sizeof(int*));
    if (array == NULL)
        return -1;
    printf("input the array\n");
    for (i = 0; i < len; i++)
        scanf("%d", array[i]);

    shift_func(array, len, k);

    printf("after shift the array is:\n");
    for (i = 0; i < len; i++)
        printf("%d", array[i]);
    printf("\n");

    return 0;
}
```

修改后的程序如下所示

```
void shift_func(int *array, int len, int k)
{
    int i = 0, j = 0;
    int temp = 0;
    if (array == NULL)
        return;
    k %= len;
    for (i = 0; i < k; i++)
    {
        temp = array[len - 1];
        for (j = len; j > 0; j--)
        {
            array[j] = array[j - 1];
        }
        array[0] = temp;
    }
}
int main()
{
    int len = 0, k = 0, i = 0;
    int *array = NULL;

    scanf("%d%d", &len, &k);
    array = (int*)malloc(len * sizeof(int));//array = (int*)malloc(len *
sizeof(int));
    if (array == NULL)
        return -1;
    printf("input the array\n");
    for (i = 0; i < len; i++)
        scanf("%d", &array[i]);//scanf("%d", &array[i]);

    shift_func(array, len, k);

    printf("after shift the array is:\n");
    for (i = 0; i < len; i++)
        printf("%d", array[i]);
    printf("\n");

    // free(array);
    // array = NULL;
    return 0;
}
```

编程题

输入年,月,日,计算这一天是该年的第几天。年份符合以下两种条件的任意种即为闰年，闰年里2月会有29天：

1.年份是4的倍数但不是100的倍数。

2.年份是40的倍数

输入描述：

输入表示日期的格式: Year, Month, Day包含三个整数:年(1<=Year3000)，月(1<= Month<=12)，日(1<=Day<=31)

思路：比较容易想到的一种方法是查表。将一年中每个月份的天数放进数组中，数组下标索引即代表月份。

这里要注意闰年的处理。为了方便，我们定义两个数组，分别对应闰年的天数和非闰年的天数。再定义一个变量flag来判断是否为闰年即可。具体代码如下所示。

```
/*
 * @Description: 北京联发科嵌入式软件工程师笔试题目
 * @Version:
 * @Autor: 嵌入式与Linux那些事
 * @Date: 2021-3-15 22:24:12
 * @LastEditors: 嵌入式与Linux那些事
 * @LastEditTime: 2021-3-15 22:39:41
 */
#include<stdio.h>
//两个数组，分别存放闰年和非闰年每个月的天数，对应 1~ 12月
int b1[12]= {31,28,31,30,31,30,31,31,30,31,30,31};
int b2[12]= {31,29,31,30,31,30,31,31,30,31,30,31};

/**
 * @Description: 累加天数
 * @Param: months，月份。flag，闰年判断标志位
 * @Return: 天数
 * @Author: 公众号【嵌入式与Linux那些事】
 */
int add(int months,int flag)
{
    int i,j = 0;
    if(flag)
    {
        for(i=0; i<months; i++)
        {
            j+=b2[i];
        }
    }
    else
    {
        for(i=0; i<months; i++)
        {
            j+=b2[i];
        }
    }
    return j;
}

int main()
{
    int years,months,days;
    //设置标志位判断闰年
    int flag = 0;
```

```
    scanf("%d,%d,%d",&years,&months,&days);

    if(years>=1 && years<=3000 && months>=1 && months<=12 && days>=1 && days<=31)
    {
        if((years%4==0&&years%100!=0))
        {
            flag = 1;
        }
        printf("result is %d",add(months,flag));
    }
    else
    {
        printf("invalid parameter");
    }
    return 0;
}
```