TEST 02

说明：本次测验时间为最长 120 分钟，满分 100 分，在答题过程中可以适当使用信息检索能力寻找相关资料，使用非自己所写的代码时，请注意引用的代码量不得超过整体代码量的 50% ，并注意在引用的代码段前后用注释标注引用内容起止位置与来源。

第一部分：选择题（30分）

说明：本部分共 3 题，每题 10 分，题型为非定向选择（可能有一个或多个选项正确）。全对得 10 分，对于有多个正确选项的题目，若未全选对则每选对一个选项得 4 分，每选择一个错误选项扣 6 分。

1-1 下列变量的声明和定义方式中正确的是：

- A: `char else[10];`
- B: `float _main_location;`
- C: `int a1[3] = {'a', 2, 3};`
- D: `char n = "k";`
- E: `int c[] = {-11};`

答案：BCE

解析：

A 项数组命名错误：`else` 是保留的关键字，不能作为变量名

D 项的 `"k"` 是字符串，不能赋值给一个字符类型的变量

1-2 以下代码中语句结束时n的值为：

```
n = 1;
do
{
    for(i=n/3; i<n; i++)
    {
        if((n-i)%2==1)
        {
            n *= 2;
        }
        else
        {
            break;
        }
    }
    n++;
}
while(n <= 20);
```

- A: 19
- B: 20
- C: 29
- D: 30
- E: 以上都不是

答案：C

解析：

> 将以上程序段补全编译即可知

1-3 下列说法不正确的是:

- A: `int negative_one[10]={-1}` 的作用是创建一个值均为 -1 的有 10 个元素的数组
- B: 选择排序中寻找一个数组内最大数字的算法时间复杂度为 `O(n)`
- C: `('2'== 50 && 5 / 2 == 2.5) == 1`
- D: 长度为 `10` 的 `char` 类型数组中最多可以存下一个包含 `10` 个字母的字符串
- E: `char country[10][10] = {"china","us","uk"}` 是一种正确的二维数组声明方式

答案：ACD

解析：

```
A 项只把数组的第一个值赋为 -1 ，而其它值为 0
C '2' == 50 的值为 true(1) ，5 / 2 == 2.5 的值为 false(0) ，（真）且（假）的值为（假），即 1 && 0 == 0
D 最多可以存下一个包含 9 个字母的字符串
```

第二部分：填空题（10分）

> 说明：本部分有 3 小题共 4 空。

2-1

衡量算法运算速度高低的一个重要指标是时间复杂度，时间复杂度越高的算法运行效率越 __ ，选择排序的时间复杂度 __ 冒泡排序的时间复杂度。

答案：低 =

2-2

在一个长度为 n 的 ASCII 编码的字符串中，'\0' 符号的下标为 __ 。

答案：n

2-3 （4分）

C 语言中可以存储非负整数的数据类型有 __（至少3种）。

答案：(unsigned) int, (unsigned) long long, (unsigned) short

第三部分：简答题（10分）

> 说明：本部分共1题，按照答题内容给分

3-1

解释以下用于在包含其他符号的表达式中提取并分离加减乘除符号与数字的代码的运行原理：

```
for(i = 0; i < stren(l); i++)
{
    if (in[i] >= 48 && in[i] <= 57)
    {
        tmp = tmp * 10;
        tmp = tmp + in[i] - 48;
    }
    else if (in[i] == '+' || in[i] == '-' || in[i] == '*' || in[i] == '/')
    {
        num[p] = tmp;
        act[p] = in[i];
        tmp = 0;
        p++;
    }
}
num[p] = tmp;
```

答案：

按点给分，最高 10 分

```
for(i = 0; i < stren(l); i++)                                                    # 结束条件是字符串的长度 （1分）
{
    if (in[i] >= 48 && in[i] <= 57)                                              # 所有代表数字的符号将进入此处处理 （1分）
    {
        tmp = tmp * 10;                                                          # 经典的十进制字符串按位转换为整形 （1分）
        tmp = tmp + in[i] - 48;
    }
    else if (in[i] == '+' || in[i] == '-' || in[i] == '*' || in[i] == '/')       # 所有代表加减乘除的符号将进入此处处理 （1分）
    {
        num[p] = tmp;                                                            # 写入分类的数组（1分）
        act[p] = in[i];
        tmp = 0;                                                                 # 归零中间值 （2分）
        p++;                                                                     # 将需要写入的位置后移 （2分）
    }                                                                            # 除了以上可用的字符外的字符均不做处理直接跳过 （3分）
}
num[p] = tmp;                                                                    # 数字数量比操作符多 1 个，需要将最后一个中间值中未储存的数字储存起来 （2分）
```

第四部分：代码题（50分）

说明：本题共两大题，第一大题30分，第二大题20分，其中第二大题难度较高，请合理安排时间。详细评分规则见题目的数据要求

4-1 奇偶排序

奇数与偶数是常见的区分整数的方法，现在给定一个整数数组 A ，要求你对其中的正奇数和正偶数分别按不同的规则进行排序

输入： 两行，第一行为一个代表数字个数的数字 n ；第二行 n 个整数

输出： 一行 n 个用空格分隔的数字，正奇数在前，按从小到大的顺序排序，正偶数在后，按从大到小的顺序排序

样例输入 1：

```
5
1 2 6 4 3
```

样例输出 1：

```
1 3 6 4 2
```

样例输入 2：

```
10
1 2 3 4 5 6 7 8 9 0
```

样例输出 2：

```
1 3 5 7 9 8 6 4 2
```

数据说明：

- n < 1000
- 输入的数字小于 ((1<<32 - 1)
- 共 10 组测试数据，每组满分为 3 分，只有输出完全正确才得分

答案：见 `OddEvenSort.c`

解析：

知识点：排序，数组

难点：排序算法

易错点：储存数字的时候要注意筛去非正数，并使用 unsigned int 或 long long 数据类型储存

4-2 睿智的敏感词过滤器

你现在需要学习微博等 APP 做一个睿智的敏感词过滤器，将一段聊天室中的对话中的敏感词全部替换成相应数量的 * 号，并封禁那个说了最多敏感词的家伙

输入： n + m + 2 行：

```
- 第一行一个数字 n ，表示敏感词的数量
- 后面 n 行每行一段不带空格的字符串，表示 n 个需要过滤的敏感词
- 下一行一个数字 m ，表示聊天的数量
- 后面 m 行每行一段对话，格式为："(ID): (content)" ，对话内容中仅含英文字母，下划线与数字
```

输出： m + 1 行：

```
- m 行过滤后的对话
- "*system*: " + 最后一行输出对话中出现敏感词最多的ID + " is banned for talking aggressively"
```

样例输入 1：

```
4
wdnmd
cnm
fuck
shit
10
A: There_will_be_a_test_today
B: fuck
A: Dont_worry_the_questions_will_be_fucking_ez
B: wdnmd_but_I_havent_even_review_anything
B: cnm
A: anyway_that_wont_be_a_hard_task_for_me
B: damn_it
A: take_it_easy
A: you_will_eventually_pass_it
B: okay
```

样例输出 1：

```
A: There_will_be_a_test_today
B: ****
A: Dont_worry_the_questions_will_be_****ing_ez
B: *****_but_I_havent_even_review_anything
B: ***
A: anyway_that_wont_be_a_hard_task_for_me
B: damn_it
A: take_it_easy
A: you_will_eventually_pass_it
B: okay
*system*: B is banned for talking aggressively
```

数据说明：

- n < 20 m < 200 ID 数量 < 10
- ID 长度 < 10 字符，敏感词长度 < 10 字符
- 每人说的内容 < 128 字符
- 共 4 组测试数据，每组满分 5 分
- 对于每组测试数据，输出每有一行出现错误则从满分扣除 2 分，扣完为止

题目提示：

- 除了通过循环查找敏感词的方法外，你还可以考虑使用 `string.h` 中的 `strstr()` 方法
- 相关链接

答案：见 `WebCleaner.c`

解析：

知识点：循环，字符串处理

难点：题目内容较多