A - 程设基础知识选择题 2

答案为 ACBDC。

第一题

- B. strlen函数的返回值是**unsigned int**型的
- C. scanf也有可能产生字符串储存溢出的风险
- D. 伪随机数的最大值可以通过修改宏定义来改变

第二题

- A. EOF和NULL是两个由编译器设定的宏，并非C语言的保留关键字
- B. 头文件可以包含多次
- C. 同正常变量命一样指令名只能以字母打头

第三题

- B. p是一个指向一维数组的指针

第四题

- A. else会与最近的一个if对应而不受到缩进的影响，这里的else对应的if错误，前半部分应该使用大括号包裹
- B. 在存在continue时，while中的i++可能无法执行，而for中的会仍然执行
- C. do-while只能保证至少执行一次，在多次执行时，它与while的执行次数往往相同

第五题

- A. 虽然会导致信息丢失，但是语法上时合法的，有时通过这种方法保留特定的数据
- B. x必须是整型变量才可以进行位运算的操作
- D. void变量不能被以任何方式转换成非空类型

B - GXY_FOREVER_LOSE

知识点	难度
循环	1

题目分析

循环输出从 a 到 b 的所有整数即可。

样例代码

```c
#include <stdio.h>

int main()
{
    int a, b;

    scanf("%d%d", &a, &b);
    while (a <= b)
    {
        printf("%d ", a++);
    }

    return 0;
}
```

C - Chainsaw

难度	考点
1	数组、循环

题目分析

按照题目要求，一个数字是 Chainsaw 数，当且仅当其比数组两端的数字都大。所以只需要循环数组一遍，根据要求判断每个数字是否满足条件，随后计数，即可获得最终答案。

示例程序

```c
#include <stdio.h>

int n, ans;
int a[1100];

int main()
{

    scanf("%d", &n);

    for (int i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    for (int i = 0; i < n; i++)
    {
        if (a[i] >= a[0] && a[i] >= a[n - 1])
        {
            ans++;
        }
    }

    printf("%d", ans);

    return 0;
}
```

D - 超级马外奥!

知识点	难度
循环	2

题目分析

本题只需按照题目中的要求进行模拟即可。

样例代码

```c
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);
    int cnt = 0;
    for (int i = 1; i <= n; i++)
    {
        int x;
        scanf("%d", &x);
        cnt += x + 1;
    }
    printf(cnt >= 50 ? "GUGUGU !" : "Yes we can !");

    return 0;
}
```

E - 修草坪

知识点	难度
循环、二维数组	3

题目分析

利用行数奇偶性判断，用循环模拟输出即可。注意循环变量范围的边界和循环变量变化的方向。

样例代码

```c
#include <stdio.h>

int main()
{
    int n, m, a[20];

    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            scanf("%d", &a[j]);
            if (i % 2 == 1)
                if (a[j] != 0)
                    printf("%d ", a[j]);
        }
        if (i % 2 == 0)
            for (int j = m; j >= 1; j--)
                if (a[j] != 0)
                    printf("%d ", a[j]);
    }

    return 0;
}
```

F - 坏掉的键盘（们）

知识点	难度
数组	3

题目分析

本题与题目 `Quick Brown Fox`（题目ID: 3824）相似。分别使用两个数组存储是否出现过相应字符，最后再按题目要求顺序输出没有出现过的即可（或输出相应字符串）。

注意点：

- 大小写字母均应该视为相应按键是好的；
- 在处理新的一组数据时，注意初始化相应变量与数组；
- 注意使用 `gets` 函数时，适当处理第一行输入数字后面的换行（建议用 `scanf("%d ")` 的形式预先将多余的换行读取）。

样例代码

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
int main() {
    int alpha[26] = {0}, digit[10] = {0};
    char sen[105] = {0};
    char *t;
    int i, j;
    int n;
    scanf("%d ", &n);
    for (; n; n--) {
        gets(sen);
        for (i = 0; i < 26; i++) {
            alpha[i] = 0;
        }
        for (i = 0; i < 10; i++) {
            digit[i] = 0;
        }
        for (t = sen; *t != '\0'; t++) {
            if (isdigit(*t)) {
                digit[*t - '0']++;
            }
            if (isupper(*t)) {
                alpha[*t - 'A']++;
            }
            if (islower(*t)) {
                alpha[*t - 'a']++;
            }
        }
        j = 0;
        for (i = 0; i < 10; i++) {
            if (digit[i] == 0) {
```

```
                printf("%c ", i + '0');
                j = 1;
            }
        }
        for (i = 0; i < 26; i++) {
            if (alpha[i] == 0) {
                printf("%c ", i + 'a');
                j = 1;
            }
        }
        if (j == 0) {
            printf("Zzz LOVES this keyboard!");
        }
        puts("");
    }
    return 0;
}
```

G - bignum_popcount()

考点	难度
高精度除法	3

题目分析

将高精度十进制转二进制后，统计二进制中 1 的个数即可完成本题，思路可参考 C7-H。

示例代码

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

#define N (1000 + 5)

int IsZero(int a[], int n)
{
    for (int i = 1; i <= n; i++)
        if (a[i]) return 0;
    return 1;
}

char s[N];
int a[N];

void Solve()
{
    scanf("%s", s);
    int n = strlen(s);
    for (int i = 1; i <= n; i++)
        a[i] = s[n - i] - '0';

    int ans = 0;
    while (!IsZero(a, n))
    {
        ans += a[1] & 1;
        for (int i = n; i > 0; i--)
        {
            a[i - 1] += a[i] % 2 * 10;
            a[i] /= 2;
        }
    }
    printf("%d\n", ans);
}

int main()
{
    int t;
    scanf("%d", &t);
    while (t--) Solve();

```

```
42      return 0;
43  }
```

H - 体能锻炼

考点	难度
排序	3

题目分析

1. 首先，按照时间线每过一秒调整每个同学的状态的做法，可以得到一部分分数，但是会 `TLE`，因为根据数据范围，$T \leq 10^4 \quad n \leq 10^4$，有 $T \times n \leq 10^8$，即最多可能进行 10^8 次运算，在时间限制为 1s 时，`TLE` 时意料之中的事。
2. 读题发现，每个同学相遇时会各自转向，那么可以想到，如果忽略 TD 线的长度限制，无论经过多长时间，每个同学的相对位置是不变的。也就是说， 同学张三在初始时位于从北数第 m 个，则 T 秒之后，他的位置还是在从北数第 m 个。这个特点能帮我们知道 T 秒之后的 n 个位置中 “谁是谁”。
3. 剩下的问题是怎么确定 T 秒之后的 n 个位置。因为只需要求位置，所以把 n 个同学看作是无区别的人，那么可以发现，“相向而行的两个人相遇之后同时转向往回走”，看起来与“两个人相遇之后不转向擦肩而过继续走” 没什么区别。所以对于每个起始位置 x（以及其初始朝向），可以通过直接加上 T 秒的运动路程（不考虑转向），求得 T 秒后的位置 y（这两个位置 x 和 y 并不对应，但是最后得到的 n 个位置恰好是从初始位置开始 T 秒之后的变化结果）。

示例代码

```c
#include <stdio.h>
#define N 10010
int a[N][3] = {0}, b[N][3] = {0};                    // 第一列a[i][0]:位置 ; 第二列a[i][1]:输入顺序 ; 第三列a[i][2]:朝向
const char ansstr[][10] = {"N", "S", "Turning"}; // 用字符串常量简化代码
int rank(const void **a, const void **b)
{
    return (*((int *)a) - *((int *)b)); // 按照二维数组的第一列的数值从小到大排序每行
}
int id(const void **a, const void **b)
{
    return (*((int *)a + 1) - *((int *)b + 1)); // 按照二维数组的第二列的数值从小到大排序每行
}
int main()
{
    int l, n, t, i;
    char p;
    scanf("%d%d%d", &l, &n, &t);
    for (i = 0; i < n; i++)
    {
        a[i][1] = i + 1;
        scanf("%d %c", &a[i][0], &p);
        if (p == 'S')
        {
            b[i][0] = a[i][0] + t;
            b[i][2] = 1;
        }
```

```
        else if (p == 'N')
        {
            b[i][0] = a[i][0] - t;
            b[i][2] = 0;
        }
    }
    qsort(a, n, sizeof(a[0]), rank); // 把 a 数组每行按照初始位置从北至南排序
    qsort(b, n, sizeof(b[0]), rank); // 把 b 数组每行按照最终位置从北至南排序
    for (i = 0; i < n; i++)
    {
        b[i][1] = a[i][1];
        if (i < n - 1 && b[i][0] == b[i + 1][0])
        {
            b[i][2] = 2;
            b[i + 1][2] = 2;
        } // T 秒之后恰好相遇的同学方向需要单独处理
    }
    qsort(b, n, sizeof(b[0]), id); // 把 b 数组每行按照输入顺序从先到后排序
    for (i = 0; i < n; i++)
        if (b[i][0] < 0 || b[i][0] > l)
            printf("Finished\n"); // 走出 TD 线
        else
            printf("%d %s\n", b[i][0], ansstr[b[i][2]]); // 还在 TD 线上的，输出位置和朝向
    return 0;
}
```

I - 大小印第安人

难度	考点
6	数据类型，进制转换，位运算

题目分析

根据题意，我们需要判断所给的内存数据 S 是整数 N 的大端存储形式还是小端存储形式。对于给出的整数 N 我们需要通过位运算来取出 N 的每个字节，而对于二进制字符串 S 可以以每 8 位为一个单位进行进制转换得到从低地址到高地址的每个字节。随后分别进行正序与反序判断整数的每个字节和内存中相对应的字节是否一致。

由于本题中涉及到的整数最多为 8 个字节，可以直接采用 `long long` 类型的变量存储整数 N 。对于字节数低于 8 的整数，由补码的性质可知，只需从低到高取出相应个数的字节即可。

示例代码

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

typedef long long ll;

char bincode[66]; // 二进制编码
char bytes[8];    // 按顺序存储每个字节，这里用 char 代表一个字节(不表示字符)

// 提取字节：取出 8 位二进制，转化成一个 char 变量
char getByte(char *sp)
{
    int i;
    char ret = 0;
    for (i = 0; i < 8; i++)
    {
        ret = (ret << 1) + (sp[i] - '0');
    }
    return ret;
}

// 字节数组的反转
void reverse(char *left, char *right)
{
    char tmp;
    while (left < right)
    {
        tmp = *left;
        *left = *right;
        *right = tmp;
        left++;
        right--;
    }
}

// 判断整数和字节数组是否匹配
```

```c
int check(int size, ll N, char bytes[])
{
    int i;
    for (i = 0; i < size; i++)
    { // 从低到高取出 N 的各个字节
        // 采用位运算取出整数 N 的从低到高第 i 个字节(从 0 开始)
        char byte_from_N = (N & (0xFFLL << (i * 8))) >> (i * 8);
        if (byte_from_N != bytes[i])
            return 0; // 不匹配
    }
    return 1; // 匹配
}

int main()
{
    int T, b, i;
    ll N;
    int bigEndian, littleEndian;
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d %lld %s", &b, &N, bincode);
        for (i = 0; i < b; i++)
        {
            bytes[i] = getByte(bincode + (8 * i));
        }
        littleEndian = check(b, N, bytes); // 判断是否为小端存储
        reverse(bytes, bytes + b - 1);
        bigEndian = check(b, N, bytes); // 判断是否为大端存储

        if (bigEndian && littleEndian) // 二者都满足，所以无法判断
            puts("???");
        else if (bigEndian)
            puts(">>>"); // 大端存储
        else if (littleEndian)
            puts("<<<"); // 小端存储
        else
            puts(">_<");
    }

    return 0;
}
```

J - du shu zi (加强版)

考点	难度
字符串处理	4

题目分析

为简化代码，预先将每一位数字的读法存入字符数组，直接调用即可。

在遍历数字串的时候，遇到负号、小数点、小数点后有意义数字，直接输出读法即可；对于小数点前数字的位数进行分类讨论并**细心**处理，详细方法见示例代码。

示例代码

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

char num[15][15] = {"ling", "yi", "er", "san", "si", "wu", "liu", "qi",
"ba", "jiu"};

int main()
{
    char s[233];
    int i;
    while (gets(s + 1))
    {
        int cnt1 = 0, cnt2 = 0, len = strlen(s + 1);
        for (i = 1; i <= len; i++)
        {
            if (s[i] == '-') continue;
            if (s[i] == '.') break;
            ++cnt1; //统计小数点前数字位数
        }
        if (strchr(s + 1, '.'))
        { //搜索到小数点
            for (i = len; i != '.'; i--)
            {
                if (s[i] == '0')
                    ++cnt2; //统计小数点后无意义零位数
                else
                    break;
            }
        }
        for (i = 1; i <= len; i++)
        {
            if (s[i] == '-')
                printf("fu "); //直接读出
            else if (s[i] == '.' && len - cnt2 != i)
                printf("dian "); //小数点后含有意义数字才读出
            else
            {
                if (cnt1 == 1)
                    printf("%s ", num[s[i] - '0']); //个位直接读出 (包括 0)
```

```
                else if (cnt1 == 2)
                {
                    printf("%s shi ", num[s[i] - '0']);                 //十位直接读出
                    if (s[++i] != '0') printf("%s ", num[s[i] - '0']); //个位非零则读出
                }
                else if (cnt1 == 3)
                {
                    printf("%s bai ", num[s[i] - '0']);
 //百位直接读出
                    if (s[++i] != '0') printf("%s shi ", num[s[i] - '0']);
//十位非零则读出
                    if (s[++i] != '0')
                    {                                                   //个位非零再分类讨论
                        if (s[i - 1] == '0') printf("ling "); //十位为零，则应多读一个"ling "
                        printf("%s ", num[s[i] - '0']);       //否则直接读出
                    }
                }
                else
                {
                    if (len - cnt2 >= i) printf("%s ", num[s[i] - '0']); //读出小数点后有意义部分
                }
                cnt1 = -1; //处理完小数点前数字后将cnt1更改为 1,2,3 之外的数
            }
        }
        puts(""); //多组数据记得换行
    }

    return 0;
}
```