第 1 题：用 C 语言实现一个公用库函数 void * memmove(void *dest,const void *src,size_t n)。该函数的功能是拷贝 src 所指的内存内容前 n 个字节到 dest 所指的地址上。注意，作为公用库函数，请注意安全检查，注意处理内存区重合的情况。

第 2 题：已知一个字串由 GBK 汉字和 ansi 编码的数字字母混合组成，编写 C 语言函数实现从中去掉所有 ansi 编码的的数字和字母（包括大小写），要求在原字串上返回结果。函数接口为：int filter_ansi(char* gbk_string)。注：汉字的 GBK 编码范围是 0x8140 - 0xFEFE

第 3 题：芯片测试。有 2k 块芯片，已知好芯片比坏芯片多。请设计算法从其中找出一片好芯片，并说明你所用的比较次数上限。其中：好芯片和其它芯片比较时，能正确给出另一块芯片是好还是坏；坏芯片和其它芯片比较时，会随机的给出好或是坏。

在这里填写答案：

———

第 1 题：用 C 语言实现一个公用库函数 void * memmove(void *dest,const void *src,size_t n)。该函数的功能是拷贝 src 所指的内存内容前 n 个字节到 dest 所指的地址上。注意，作为公用库函数，请注意安全检查，注意处理内存区重合的情况。

```
void*   memmove(void * dest, const void * src, size_t n)
{
      void* temp = dest;

      if (dest <= src || (char *)dest >= ((char *)src + n)) //无内存地址重叠
      {
          while (n--)
          {
              *(char *)dest = *(char *)src;
              dest = (char *)dest + 1;
              src = (char *)src + 1;
          }
      }
      else                        //有内存地址重叠
      {
          dest = (char *)dest + n - 1;
          src = (char *)src + n - 1;
```

```
            while (n--)
            {
                *(char *)dest = *(char *)src;
                dest = (char *)dest - 1;
                src = (char *)src - 1;
             }
        }

        return (temp);
    }
```

——

第 2 题：已知一个字串由 GBK 汉字和 ansi 编码的数字字母混合组成，编写 C 语言函数实现从中去掉所有 ansi 编码的的数字和字母（包括大小写)，要求在原字串上返回结果。函数接口为：int filter_ansi(char* gbk_string)。注：汉字的 GBK 编码范围是 0x8140 - 0xFEFE

```
int filter_ansi(char* gbk_string)
{
   char *p = gbk_string, *q = gbk_string;
   while (*q != '\0')
   {
      if ((*q >= 0) && (*q <= 128))     //判断是否为 asci 的字符
      {
         if ((((*q >= '0') && (*q <= '9'))       //判断是否为数字或字母
             || ((*q >= 'a') && (*q <= 'z'))
             || ((*q >= 'A') && (*q <= 'Z')))
         {
            q++;
         }
         else
         {
            *p++ = *q++;
    }
        }
        else
        {
          if (((*((unsigned short*)q)) >= 0x8140) && ((*((unsigned short*)q)) <= 0xFEFE)) //
是汉字
           {
         *p++ = *((char*)q)++;
         *p++ = *((char*)q)++;
           }
      else       //不是汉字
```

```
        {
            q++;
            q++;
        }
            }
    }
    *p = '\0';

    return (p - gbk_string);
}
```

——

第 3 题：芯片测试。有 2k 块芯片，已知好芯片比坏芯片多。请设计算法从其中找出一片好芯片，并说明你所用的比较次数上限。其中：好芯片和其它芯片比较时，能正确给出另一块芯片是好还是坏；坏芯片和其它芯片比较时，会随机的给出好或是坏。

答案：

1．首先两个两个分成一对。如果互测的结果是好好，那么留下，否则扔掉。扔掉的里面总是坏的不会比好的少。这样剩下的要么是两个好的，要么是两个坏的。可以称剩下的这样的对为纯粹对。经过这一次分对，最坏的情况是 1000 对都留下；

2．然后把留下的对再随便两两分组。这样变成每组里有两个纯粹对。每组比较时，从这两个纯粹对中任意各选一个元素进行比较。如果都是好的，那么留下这组，否则扔掉。这样剩下的要么四个都是好的，要么四个都是坏的，我们把四个看成一个组，这个组是纯粹组。经过这一次分对，最坏的情况是 500 对都留下；

3．接着把剩下的这些组再随便两两分对。得到的每对里有两个纯粹组。比较每对里的两个纯粹组时，还是任意各选一个元素比较。如果都是好的，那么留下这对，得到一个新的纯粹组，里面有 8 个好的或者 8 个坏的。否则扔掉。最坏的情况是留下 250 对；

4．再把这些剩下的组两两分对。同样从每对的两个纯粹组中各任选一个元素进行比较。都是好的留下，成为新的纯粹组。否则扔掉。这样可得到一个新的纯粹组，其中 16 个都是好的或者 16 个都是坏的。最坏的情况是留下 125 对；

5．同样两两分对，得到 62 对和一个单独的。我们用同样方法比较这 62 对。最坏的情况是留下 31 对，其中每对里 32 好或者 32 坏，和单独的一个，其中 16 好或者 16 坏。把它们带入下一轮；

6．把剩下的 32 对再任意两两分组，其中一组是一个含有 32 好或坏，另一个含有 16 个好或坏，记为 A。其它的 15 组都是两个含有 32 好或坏的。同样方法比较这 15 组，得到新的纯粹组，含有 64 好或 64 坏。最坏的情况留下 15 组。至于 A, 我们也是两个中各任选一个代表比较，如果同好，那么留下，成为新的纯粹组，含有 48 个好或者坏。否则从含有 32 好或坏的那组中任意选 16 个和另一组含有 16 个好或坏的一起扔掉，剩下的 16 个同好或同坏的

成为一个新的纯粹组。这样我们可以保证扔掉的里面总是坏的不比好的少。也就是说最坏的情况是剩下 16 组。

7．把这 16 组两两分对。用和第六步一样的方法。我们最坏可以得到 8 个。其中 7 个是 128 好或者 128 个坏，还有一个是数目小于 128 的纯粹组。

8．同理，再两两分对，我们可以得到 4 组；然后得到两组。这时只要从数量最多的那个组中任选一个，即为好的。所以最坏的情况是要比较 1000+500+250+125+62+31+16+8+4+2=1998 次. （如果剩下的是两组芯片个数一样多，那么可以从两个组中任选一个，即为好的。因为好的数量总比坏的数量多，而且我们每次扔掉的都是坏的比好的多或者相等。所以剩下的这两个组中的芯片只可能都是好的。）