Assignment1 document

一．问题描述及规格说明（需求分析）

返回长度为 n 的所有非负整数，使每两个连续数字之间的绝对差为 k。请注意，答案中的每个数字都不能有前导零。并且需要按照大小进行输出。

二．设计（目标：有效的组织和处理数据）

数据结构设计：根据两个输入参数：数字长度(n),相邻两个数字绝对差(k),我们无法通过简单的计算来得到总共有多少个符合要求的数字。所以我想到了利用可变长度的数组也就是 vector 来存储数字，而且 vector 有对应封装好的函数使我们方便进行增删，正好满足我设计算法中需要频繁对数组的 end 位置进行操作（增加，删减）的需求。

算法类的设计：本次算法类包含四个数据成员，一个静态成员变量 sign 用来记录输出符合要求数字的个数，两个成员变量分别是 size（数字长度），differ_ num(相邻两个数字绝对差)。以及一个公有成员数据 vector<int> buf。成员函数是为了实现基本功能。

三．算法设计

先约定，对符合要求的数（也就是要输出的数）记作 correct _num,对于符合要求的数的每一位数记作 num,对于相邻两个数的绝对差记作 differ _num，对于符合要求的数的位数记作 num _count,

基本想法：注意题目的条件，数字的位数 n（n>=1）从键盘上输入，所以 n 很有可能会很大（一百甚至上千）。此时不能把符合条件的数字当作是 n 位数字，而应该把数字拆分成一个个位数。但由于拆分过程十分繁琐，所以逆向思维把每个 num 求出来，存进数组 buf 里面，之后输出就可以组合成 correct _num，并添加逗号作为分隔。

设置一个函数 function(int num ,int num_ count), 包含两个参数:每位数字的大小 num（0<=num<=9）, 数字的位数 num_ count（即每选取一个符合要求的 num，数字位数 num _count 会加 1）。

函数内部对于 num 可以进行加法减法操作:如 num+ differ_ num 或者 num-differ_ num，如果满足（num - differ_ num>=0 || num + differ_ num<=9）就可以存进数组 buf 里面,接下来又会对新产生的两个数执行函数的功能。所以我考虑递归，对数来说，第 k 位到第 n 位数字（1<= k<=n）都满足于 function(int num ,int num_ count)，所以在函数参数中 num 会被替换成 num - differ_ num 或者 num + differ_ num。

注意 Function 函数中的两个 if 语句虽然是并列关系，但我选择了先进行减法的判断，再进行加法的判断。是因为我们关于数字 num 的筛选是对 correct _num 从最高位到最低位进行的，所以要保证最后的输出结果是升序排列，必须先进行减法判断，使得高位尽可能的小。仍然需要注意 k=0 这种特殊情况，因为此时 num-differ _num=num +differ _num,会导致重复计算，所以在第二次 if 语句加法判断的时候，务必加上 differ _num!=0 这个条件。

递归的终止条件是：当数字的总个数也即是 num_ count== n 个，就把 buf 数组里面所有的数据从 buf .front() 到 buf .end()输出。接下来进行递归

回溯，来寻找下一个符合条件的 correct _num，以此类推。

四. 模块及调用关系

主函数做的是把题目要求的两个参数输入，并对其错误输入做一些简单的判断。这两个参数会作为 solution 类的两个参数来构造对应的实例对象。并且通过构造函数会初始化两个数据成员变量 size 和 differ _num ；size=n, differ _num=k。

Solution 类的函数入口为 main_ function(),进行一个九次的 for 循环，循环内部调用了 function(int num, int num _count)这个函数，也就是说把所有符合要求的数字，按照最高位的数字 num（1-9）进行划分，然后分别进行求解答案。

Display()函数是用来输出 buf 里面的数据的。注意问题求解的过程是递归的，function()函数每次递归结束的时候，会执行一次 display()的输出。

```
class solution
{
private:
    static int sign;//标志位用来标记输出的是否是第一个数字，若是则不需要输出逗号
    int size;//用来表示vector数组的大小,也等于数字的位数
    int differ_num;//表述相邻两位差的绝对值
public:
    vector<int>buf;//用来存储数字
    solution(int  x = 0, int k = 0) :size(x), differ_num(k){};//构造函数
    void function(int num,int num_count);//n代表数字位上的大小，num_count为记录数字的位数
    void display();//输出数组中的元素
    void main_function();//k为相邻两位差值的绝对值
    ~solution() {};//析构函数
};
```

五. 算法复杂度分析

空间复杂度：能达到比较好的效果，由于只创建了一个数组 buf,大小为输入参数 n，所以空间复杂度是 O(n)。

时间复杂度：function()函数中有递归的形式，而且拥有两个 if 语句，经过分析可得到，当 k>=5 and k==0 的时候，由于 num-differ _num 和 num+ differ _numd 对于任意 num 只能有一个成立所以一次函数调用中，只会进入一个 if 子句；而当 0<k<5 的时候，对于部分 num，函数两个 if 语句都会进入，而对于部分 num，函数只会进入一个 if 语句，所以需要加权计算时间复杂度，最后在 main _function()中 function()循环调用了九次，所以最终的 f(n)有下图结果可以得出

$$for\ num < k\ or\ num > 9-k$$
$$F(n)=F(n-1)+O(1)\quad n\geq 2\quad F(1)=0$$
$$for\ num \geq k\ and\ num \leq 9-k$$
$$G(n)=2G(n-1)+O(1)\quad n\geq 2\quad G(1)=2$$
$$then$$
$$F(n)=O(n)\quad G(n)=O(2^n)$$

$$if\ k<5\ and\ k>0$$
$$T_1(n)=\frac{k}{5}F(n)+\left(1-\frac{k}{5}\right)G(n)$$
$$else\ if\ k\geq 5\ and\ k=0$$
$$T_2(n)=F(n)$$

$$so$$
$$T(n)=\frac{2}{5}T_1(n)+\frac{3}{5}T_2(n)$$
$$T(n)=O(c_1kn+(1-c_2k)2^n))=O((1-ck)2^n)$$
$$c_1\ c_2\ c\ is\ constant$$
$$loop\ run\ nine\ times$$
$$the\ last\ answer\ f(n)=9T(n)=O((1-ck)2^n)$$

可以得出结论：当 k 越小，n 越大的时候，算法运行时间就会越多。

六. 功能测试

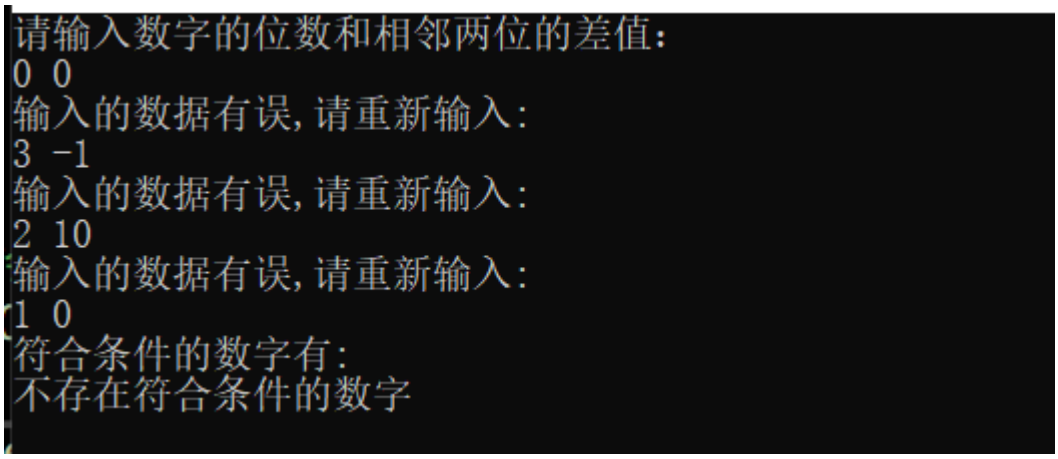
```
请输入数字的位数和相邻两位的差值：
2 1
符合条件的数字有：
10, 12, 21, 23, 32, 34, 43, 45, 54, 56, 65, 67, 76, 78, 87, 89, 98
```

```
请输入数字的位数和相邻两位的差值：
3 7
符合条件的数字有：
181, 292, 707, 818, 929
```

```
请输入数字的位数和相邻两位的差值：
2 0
符合条件的数字有：
11, 22, 33, 44, 55, 66, 77, 88, 99
```

七. 出错测试

边界测试：输入的 n 和 k 必须满足条件(n>0&&0<=k<=9)



错误输入测试：输入的数据不是正整数