ACM ICPC2014 国内予選　問題 D 暗号化システム

【考察】

- 文字列の長さは 20 文字以下であり、暗号化された各文字に対する暗号化前の文字は 2 通り（同じ文字か一つ後の文字）以下なので、元の文字列の候補数は高々2^{20} ≒ 10^6 なので、これらを順に暗号化してみて暗号化後の文字列と一致するかを判定すれば良い。
- 元の文字列の候補としては、深さ優先探索（左の子→暗号化した文字、右の子→暗号化した文字の次の文字）で探索することにより、元の文字列候補を辞書式にチェックすることができる。
- 暗号化文字列の先頭の k 文字は、元の文字列の k+1 文字以降の影響は受けないので、先頭の k 文字を暗号化してみて候補とならないことが判明した場合は、k+1 文字以降のチェックは打ち切ることができる（k > 0）。
- 候補が見つかった場合、最初の 10 個までは文字列の配列に先頭から順に格納し、10 個を超えたら、文字列の配列の 5 番目から 9 番目までをリングバッファのようにサイクリックに書き込むことで、末尾の 5 個の文字列を覚えておくことができる。

【プログラム例】

```
// ACM-ICPC2014 Domestic Problem D 暗号化システム
//   http://icpc.iisf.or.jp/past-icpc/domestic2014/#section_D
//   Filename =      pd2.c
//   Compile:        cc pd2.c
//   Execution:      ./a.out < D0 > D0.out
//   Check:          diff C0.ans C0.out
//   Method:         元の文字列の候補 2^len 通りを深さ優先探索でチェック
//                   (len = 暗号化文字列の長さ)
//                   但し、先頭の k 文字で候補にならないことが判明したら
//                   k+1 文字以降の探索は打ち切る (k > 0)

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

#define LEN 20                  // 文字列の最大長
char pat[LEN+1];                // 暗号化文字列
char str[10][LEN+1];            // 答えの暗号化前文字列
                                // 候補数が 10 以下の場合、str[0]～str[count-1]に格納
                                // 候補数が 11 以上の場合、str[0]～str[4]に最初の 5 候補、
```

```
                    // str[5]～str[9]はリングバッファとして用い、
                    // 末尾の 5 候補を格納
int lp;             // 答えの末尾 5 個の開始位置（リングバッファの先頭）
int count;          // 元の文字列の候補数

// 文字列 s の先頭 k 文字が暗号化前の文字列の候補なら 1 を、
// そうでなければ 0 を返す
int check(char *s, int k)
{
  int i,j;
  char w[LEN+1];            // 作業用文字列領域
  strcpy(w,s);              // 作業用領域にコーピー
  // 文字列の最初の k 文字に暗号化ステップ 1～25 を適用
  for(i=1; i<=25; i++){
    for(j=0; j<k; j++){                 // 先頭から k 文字の各文字にステップ i を適用
      if(w[j] == 'z'+1) return 0;       // 'z'の後の文字を含んではダメ
      if(w[j] == 'a'+i){                // 文字が'a'+i ならその文字を'a'+i-1 に変更
        w[j]--;
        break;                          // 変更するのは最初の文字だけ
      }
    }
  }
  // 文字列 w の先頭 k 文字は暗号化後の文字列と一致するか？
  for(j=0; j<k; j++)
    if(w[j] != pat[j]) return 0;        // 一致しなければ候補ではない
  return 1;                             // 先頭 k 文字に関しては候補である。
}

// 深さ優先探索で、候補文字列を探す。
// 先頭の d 文字で候補になり得ないことが分かれば、その下の探索は打ち切る
void search(char *s, int d){
  char t[LEN+1];
  int ret;
  ret = check(s, d);        // 文字列 s の先頭 d 文字は候補になり得るか？
  if(ret==0) return;        // なり得ないので、以降の探索は打ち切る
  if(s[d] == 0){            // s[d] == 0 means d == len. => 候補であることが判明
    if(count<10){           // これまでに見つかった候補数 count が 10 未満なら
      strcpy(&str[count][0],s);         // count 番目に文字列を格納
    }
    else {                  // 既に 10 個以上候補が見つかっている場合は、
                            // 最後から 5 番目に見つかった文字列の場所に格納
      strcpy(&str[lp++][0],s);
      if(lp >= 10) lp = 5;  // 最後の 5 個の文字列の先頭位置を更新
    }
    count++;                // 見つかった候補数を 1 増やす
  }
  else {                    // 文字列の途中を探索中
```

```
    search(s, d+1);      // d 番目の文字は変化しないと仮定して d+1 番目以降を探索
    if(s[d] != 'z'){          // d 番目の文字が'z'でなければ
      s[d]++;                 // この文字は暗号化ステップで変化したと仮定し
      search(s, d+1);         // d+1 番目以降を探索
      s[d]--;                 // d 番目の文字を変化前に戻す
    }
  }
}

int main()
{
  int i,j,k;
  char work[LEN+1];                  // 作業用文字列領域
  while(1){
    scanf("%s", pat);                // 暗号化後の文字列の入力
    if(strcmp(pat,"#")==0) break;    // "#"なら終了
    count = 0;                       // 候補数をリセット
    lp = 5;                          // 末尾 5 個の開始位置：最初は 5
    strcpy(work, pat);               // 作業用領域にコピー
    search(work, 0);                 // 候補の探索
    printf("%d¥n",count);            // 候補数を出力
    if(count<=10)                    // 候補数が 10 以下の場合は、
      for(i=0; i<count; i++)         // 先頭から count 個候補文字列を出力
        printf("%s¥n", &str[i][0]);
    else {                           // 候補数が 11 以上の場合
      for(i=0; i<5; i++)             // まず、最初の 5 候補を出力
        printf("%s¥n",&str[i][0]);
      for(i=0; i<5; i++){            // 次に末尾の 5 個の候補を
        printf("%s¥n", &str[lp++][0]);        // 開始位置 lp から順に 5 個出力
        if(lp >= 10) lp = 5;
      }
    }
  }
}
```