次に示すポインタ版の「Boyer-Moore 法による文字列探索」プログラムを入力し，自分のパソコンでコンパイル，実行できることを確認してください．なお，プログラムの日本語部分は，英語，ローマ字に変更してかまいません．さらに，以下の問いに答えなさい．

```
#include  <stdio.h>
#include  <string.h>
#include  <limits.h>
/*--- Boyer-Moore 法による文字列探索 ---*/
char *bm_match(char *pat , char *txt){
  char   *pt;                   /* txt をなぞるカーソル */
  char   *pp;                   /* pat をなぞるカーソル */
  int    txt_len = strlen(txt); /* txt の文字数 */
  int    pat_len = strlen(pat); /* pat の文字数 */
  int    skip[UCHAR_MAX + 1];   /* スキップテーブル */
  int    i;

  for (i = 0; i <= UCHAR_MAX; i++) /* スキップテーブルの作成 */
    skip[i] = pat_len;
  for (pp = pat; *pp != '¥0'; pp++)
    skip[*pp] =  strlen(pp) - 1;
  skip[*(pp - 1)] = pat_len;  /* パターンの最後文字の移動距離はパターンの文字数 */
  pt = txt + pat_len - 1;       /* pat の末尾と比較する txt の文字を決定 */
  while ( pt < txt + txt_len) { /* txt の比較する文字の位置が txt の末尾を越えるまで */
    pp = pat + pat_len - 1;     /* pat の最後の文字に着目 */
    while (*pt == *pp) {        ①
      if (pp == pat) return (pt); /* 一致した文字がパターンの最初の文字になれば終了 */
      pp--;
      pt--;
    }
    pt += (skip[*pt]> strlen(pp)) ? skip[*pt] : strlen(pp);
  }
  return (NULL);
}
int main(void){
  char  *s;
  char  s1[80]; /* テキスト */
  char  s2[80]; /* パターン */

  printf("テキスト:");
  scanf("%s", s1);
  printf("パターン:");
  scanf("%s", s2);
  s = bm_match(s2, s1); /* 文字列 s1 から文字列 s2 を Boyer-Moore 法で探索 */
                                                          ②
  if (s == NULL)
    puts("テキスト中にパターンは存在しません。");
  else
    printf("%d 文字目に見つかりました。¥n", s - s1 + 1);
  return (0);
}
```

1）キーボードからテキストとして「CAGACAGAGA」を，パターンとして「AGAG」を入力したとき，次の問に答えなさい．

(ア) スキップテーブルを表す配列の skip['G']と skip['A']の値はいくらですか．

skip['A'] = 1, skip['G'] =4

(イ) 下線②の関数 bm_match(s2, s1)の戻り値のポインタが指している文字は，テキスト「CAGACAGAGA」のどの文字ですか．

CAGACAGAGA

2）キーボードからテキストとして「APCPAACBABEAAA」を，パターンとして「AAA」を入力したとき，下線部①の while の条件式（*pt == *pp）は何度評価されますか．

①	A A	P A	1 C A	P	A	A	C	B	A	B	E	A	A	A
②	A	P	C	4 P A	3 A A	2 A A	C	B	A	B	E	A	A	A
③	A	P	C	P	A A	A A	5 C A	B	A	B	E	A	A	A
④	A	P	C	P	A	A	C	B A	A A	6 B A	E	A	A	A
⑤	A	P	C	P	A	A	C	B	A	B	9 E A	8 A A	7 A A	A A
⑥	A	P	C	P	A	A	C	B	A	B	E	12 A A	11 A A	10 A A

12 回

3）Boyer-Moore 法のアルゴリズムを変更して，指定したパターンをテキストの末尾から先頭に向かって探索する関数 char * bm_reverse_text_match(char *pat, char *txt)を作成してください．すなわち，bm_reverse_text_match では，txt が指す文字列の末尾から先頭に向かって逆順で並んでいるものをテキストとして，pat が指す文字列のパターンを探索するものとします．但し，関数 bm_reverse_text_match の引数も戻り値も，関数 bm_match と同じとします．例えば，txt が指す文字列が”ABCDEF”の場合，pat が指す文字列が”ABC”なら「テキスト中にパターンは存在しません」となるが，pat が指す文字列が”CBA”なら「4 文字目に見つかりました。」となるようにしてください．

```
/*--- Boyer-Moore 法で，指定したパターンをテキストの末尾から探索する ---*/

char *bm_reverse_text_match(char *pat, char *txt){
   char   *pt;       /* txt をなぞるカーソル */
   char   *pp;       /* pat をなぞるカーソル */
   int    txt_len = strlen(txt);   /* txt の文字数 */
   int    pat_len = strlen(pat);   /* pat の文字数 */
   int    skip[UCHAR_MAX + 1]; /*  スキップテーブル */
   int    i;

   for (i = 0; i <= UCHAR_MAX; i++) /*  スキップテーブルの作成 */
      skip[i] = pat_len;
   for (pp = pat; *pp != '¥0'; pp++)
      skip[*pp] =    strlen(pp) - 1;
   skip[*(pp - 1)] = pat_len;        /*  パターンの最後文字の移動距離はパターンの文字数 */
   pt = txt + txt_len - pat_len ;   /* pat の末尾と比較する txt の文字を決定 */

   while ( pt >= txt) {            /* txt の比較する文字の位置が txt の先頭を越えるまで */
      pp = pat + pat_len - 1;   /* pat の最後の文字に着目 */
      while (*pt == *pp) {
         if (pp == pat) return (pt); /*  一致した文字がパターンの最初の文字になれば終了 */
         pp--;
         pt++;
      }
      pt -= (skip[*pt]> strlen(pp) ) ? skip[*pt] : strlen(pp) ;
   }
   return (NULL);
}
```