

```

/* A C implementation Bickel's name comparison      */
/* algorithm, CACM 30/3 (March 1987), p. 244.     */
/* This is generic C code and should work on any  */
/* compiler with no modification.                 */
/* Jim Howell, March 1987.                        */

/* This code is placed in the public domain. You  */
/* free to use it in any way you see fit.         */

```

```

#include <ctype.h>
#include <stdio.h>

```

```

unsigned char  MaskArray [52] = {0, 0x40,      /* a */
                                3, 0x80,      /* b */
                                2, 0x20,      /* c */
                                1, 0x10,      /* d */
                                0, 0x20,      /* e */
                                2, 0x10,      /* f */
                                2, 0x08,      /* g */
                                1, 0x08,      /* h */
                                0, 0x10,      /* i */
                                3, 0x08,      /* j */
                                3, 0x20,      /* k */
                                1, 0x04,      /* l */
                                2, 0x04,      /* m */
                                0, 0x08,      /* n */
                                0, 0x04,      /* o */
                                2, 0x02,      /* p */
                                3, 0x10,      /* q */
                                1, 0x02,      /* r */
                                0, 0x02,      /* s */
                                0, 0x01,      /* t */
                                1, 0x01,      /* u */
                                3, 0x40,      /* v */
                                2, 0x01,      /* w */
                                3, 0x04,      /* x */
                                3, 0x02,      /* y */
                                3, 0x01};    /* z */

```

```

/* Demonstrate the algorithm with some short examples. */
main ()
{
    unsigned char  LetterSet0 [4],
                  LetterSet1 [4];

    MakeLetterSet ("ecdysiast", LetterSet0);
    MakeLetterSet ("ecstasy", LetterSet1);
    printf ("The likeness of 'ecdysiast' and 'ecstasy' is %d.\n",
           CompareSets (LetterSet0, LetterSet1));

    MakeLetterSet ("ectoplasm", LetterSet1);
    printf ("The likeness of 'ecdysiast' and 'ectoplasm' is %d.\n",

```

```
CompareSets (LetterSet0, LetterSet1));
```

```
MakeLetterSet ("edcysiast", LetterSet1);  
printf ("The likeness of 'ecdysiast' and edcysiast' is %d.\n",  
    CompareSets (LetterSet0, LetterSet1));  
} /* main */
```

```
/* Make the letter set by going through the string */  
/* one character at a time. */
```

```
MakeLetterSet (Name, Mask)
```

```
char *Name;  
unsigned char *Mask;
```

```
{  
char *pC;  
unsigned char *pM;  
int l,  
    Lk;
```

```
pC =Name;
```

```
for (l =0; l <4; ++ l)
```

```
    Mask [l] =0;
```

```
while (*pC) {
```

```
    /* Use letters only, and convert */
```

```
    /* uper to lower case. */
```

```
    if (isalpha (*pC)) {
```

```
        pM =&MaskArray [2 *(tolower (*pC) -'a')];
```

```
        Mask [*pM] |=*(pM +1);
```

```
    }
```

```
    ++ pC;
```

```
}
```

```
} /* MakeLetterSet */
```

```
/* This particular version constructs a mask by using */  
/* a logical AND of the relevant bytes of the two sets. */  
/* The rightmost bit is checked to see if the likeness */  
/* score needs to be increased by the appropriate */  
/* weight, and the mask is shifted right one bit. An */  
/* alternative would be to use a bit mask and to */  
/* shift it instead of the name mask. The two bytes */  
/* of the least common letters are checked for content */  
/* to eliminate any unnecessary calculations. */
```

```
CompareSets (Set1, Set2)
```

```
unsigned char *Set1,  
             *Set2;
```

```
{  
unsigned char Mask;  
int l,  
    Lk;
```

```
Lk =0;
```

```
/* For the first byte. */
```

```

Mask =Set1 [0] & Set2 [0];
for (l =0; l <7; ++ l)  {
    if (Mask & 0x01)
        Lk +=3;
    Mask >>=1;
}

/* The second byte. */
Mask =Set1 [1] & Set2 [1];
for (l =0; l <5; ++ l)  {
    if (Mask & 0x01)
        Lk +=4;
    Mask >>=1;
}

/* The third byte. */
Mask =Set1 [2] & Set2 [2];
if (Mask)
    for (l =0; l <6; ++ l)  {
        if (Mask & 0x01)
            Lk +=5;
        Mask >>=1;
    }

/* The last byte is more complicated. */
Mask =Set1 [3] & Set2 [3];
if (!Mask)
    return (Lk);
if (Mask & 0x01)
    Lk +=9;          /* z */
Mask >>=1;
if (Mask & 0x01)
    Lk +=8;          /* y */
Mask >>=1;
if (Mask & 0x01)
    Lk +=8;          /* x */
Mask >>=1;
if (Mask & 0x01)
    Lk +=8;          /* j */
Mask >>=1;
if (Mask & 0x01)
    Lk +=7;          /* q */
Mask >>=1;
if (Mask & 0x01)
    Lk +=7;          /* k */
Mask >>=1;
if (Mask & 0x01)
    Lk +=6;

```