

1. Longest common subsequence. This is a straight-from-the-book implementation of an algorithm to determine a longest common subsequence of two given sequences.

```

< Header files to include 2 >
< Global variables 3 >
< Functions 5 >
< The main program 4 >

```

2. We must include the standard I/O definitions.

```

< Header files to include 2 > ≡
#include <stdio.h>
#include <string.h>

```

This code is used in section 1.

3. In this program, we only consider short strings.

```

#define MAXLEN 50
< Global variables 3 > ≡
    int c[MAXLEN][MAXLEN];

```

This code is used in section 1.

4. Now we come to the general layout of the *main* function.

```

< The main program 4 > ≡
    main(argc, argv)
        int argc;    /* the number of arguments on the UNIX command line */
        char **argv; /* the arguments themselves, an array of strings */
    { < Variables local to main 6 >
      < Scan the command-line options 7 >
      < Compute the length of a longest common subsequence 8 >
      < Print a longest common subsequence 9 >
      return 0;      /* Normal exit */
    }

```

This code is used in section 1.

5. The algorithm is implemented through two functions.

⟨ Functions 5 ⟩ ≡

```

void lcs_build_c(char *x, char *y, int m, int n)
{
    int i, j;
    for (i = 0; i ≤ m; i++) c[i][0] = 0;
    for (j = 0; j ≤ n; j++) c[0][j] = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (x[i] ≡ y[j]) c[i + 1][j + 1] = c[i][j] + 1;
            else if (c[i][j + 1] > c[i + 1][j]) c[i + 1][j + 1] = c[i][j + 1];
            else c[i + 1][j + 1] = c[i + 1][j];
}

void print_lcs(char *x, char *y, int i, int j)
{
    if ((i ≥ 0) ∧ (j ≥ 0))
        if (x[i] ≡ y[j]) {
            print_lcs(x, y, i - 1, j - 1);
            printf ("%c", x[i]);
        }
        else if (c[i][j + 1] > c[i + 1][j]) print_lcs(x, y, i - 1, j);
        else print_lcs(x, y, i, j - 1);
}

```

This code is used in section 1.

6. The local variables will just be the strings *x* and *y* to be compared.

⟨ Variables local to *main* 6 ⟩ ≡

```

char x[MAXLEN], y[MAXLEN];
int m, n;

```

This code is used in section 4.

7. We read the strings to compare here.

⟨ Scan the command-line options 7 ⟩ ≡

```

x[0] = y[0] = '\0';
while (--argc) {
    if (sscanf(argv[argc], "-x%s", x) ≡ 1) ;
    else if (sscanf(argv[argc], "-y%s", y) ≡ 1) ;
    else {
        printf ("Usage: _%s_ [-x<string>] [-y<string>]\n", argv[0]);
        return -1;
    }
}

```

This code is used in section 4.

8. The call to *lcs_build_c* is this.

⟨ Compute the length of a longest common subsequence 8 ⟩ ≡

```

m = strlen(x);
n = strlen(y);
lcs_build_c(x, y, m, n);

```

This code is used in section 4.

9. The printing procedure is simple as well.

⟨Print a longest common subsequence 9⟩ ≡

```
printf("A longest common subsequence of the strings\n%s\n%s\nis\n", x, y);  
print_lcs(x, y, m - 1, n - 1);  
printf("n");
```

This code is used in section 4.

10. Index. Here is a list of the identifiers used, and where they appear. Underlined entries indicate the place of definition.

argc: 4, 7.

argv: 4, 7.

c: 3.

i: 5.

j: 5.

lcs_build_c: 5, 8.

m: 5, 6.

main: 4.

MAXLEN: 3, 6.

n: 5, 6.

print_lcs: 5, 9.

printf: 5, 7, 9.

sscanf: 7.

strlen: 8.

x: 5, 6.

y: 5, 6.

- ⟨ Compute the length of a longest common subsequence 8 ⟩ Used in section 4.
- ⟨ Functions 5 ⟩ Used in section 1.
- ⟨ Global variables 3 ⟩ Used in section 1.
- ⟨ Header files to include 2 ⟩ Used in section 1.
- ⟨ Print a longest common subsequence 9 ⟩ Used in section 4.
- ⟨ Scan the command-line options 7 ⟩ Used in section 4.
- ⟨ The main program 4 ⟩ Used in section 1.
- ⟨ Variables local to *main* 6 ⟩ Used in section 4.