Representation:
A C-string is an array of characters terminated by a null character. For example, the string “toy” is represented as shown in figure 1.

```
Figure 1. C-string “toy”.
```

Declaration:
A C-string is declared as an array of characters. Examples are given below.

```c
char s[10]; //s is an array of 10 characters having elements s[0] .. s[9].
//s can hold up to 9 characters
char t[]="one";  //t is a string initialized to the characters ‘o’, ‘n’, ‘e’, ‘\0’
char u[3]="one";  //t is initialized to the characters ’o’, ’n’, ’e’
//t is not a string.
char v[]={'o','n','e','\0'}; //v is a string having four (4) characters. Each character is initialized.
char e[]="";   //e is a string having a single character, the null terminator.
//e is the empty string
```

Strings and pointers to strings:
1. Strings are referenced by pointers to the actual string. For example, variable t, is used to reference string t declared as char t="toy";
2. When the name of an array appears without a subscripting operator [], the type of the array name is changed to a pointer to the element type. For example, t has type char* because elements of t have type char and t is an array.
3. String pointers can be declared directly. For example, char* s; Variable s can be assigned to point to a string but no such assignment has been made yet. Variable s is said to be undefined. References to s will likely cause an execution-time error.
4. A string pointer can be initialized. For example char* s="toy"; Storage for string “toy” is allocated in the constant area of the program. The string “toy” cannot be changed. String s, however, can be reassigned. Refer to figure 2.

```
Figure 2. char* s="toy";
```
### Operations:

`#include <string>`

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int strlen(char* s);</code></td>
<td>Function <code>strlen</code> returns the number of characters in the string referenced by parameter <code>s</code>. The terminating character is excluded from the count returned by function <code>strlen</code>.</td>
<td><code>char s[]=&quot;one&quot;;</code>&lt;br&gt;<code>int c=strlen(s);</code>&lt;br&gt;<code>cout &lt;&lt; c;</code>&lt;br&gt;Output 3</td>
</tr>
<tr>
<td><code>char* strcpy(char* d, char* s);</code></td>
<td>Function <code>strcpy</code> copies the contents of the string <code>s</code> to the string <code>d</code>, overwriting the contents of <code>d</code>. The entire contents of <code>s</code> are copied, plus the terminating null character even if <code>s</code> is longer than <code>d</code>. The argument <code>d</code> is returned.</td>
<td><code>char d[]=&quot;destination&quot;;</code>&lt;br&gt;<code>char s[]=&quot;source&quot;;</code>&lt;br&gt;<code>char* t=strcpy(d,s);</code>&lt;br&gt;<code>cout &lt;&lt; d;</code>&lt;br&gt;Output <code>source</code></td>
</tr>
<tr>
<td><code>char* strcat(char* d, char* s);</code></td>
<td>Function <code>strcat</code> appends the contents of string <code>s</code> to string <code>d</code>. A pointer to string <code>d</code> is returned. The null character that terminates <code>d</code> (and perhaps other characters following it in memory) is overwritten with characters from <code>s</code> and a new terminating null character. Characters are copied from <code>s</code> until a null character is encountered in <code>s</code>. The memory beginning with <code>d</code> is assumed to be large enough to hold both strings.</td>
<td><code>char d[10]=&quot;One&quot;;</code>&lt;br&gt;<code>char s[]=&quot;two&quot;;</code>&lt;br&gt;<code>char* t=strcat(d,s);</code>&lt;br&gt;Output <code>One, two</code></td>
</tr>
<tr>
<td><code>int strcmp(char* u, char* v);</code></td>
<td>Function <code>strcmp</code> lexicographically compares the contents of the null-terminated string <code>u</code> with the contents of the null-terminated string <code>v</code>. It returns a value of type <code>int</code> that is less than zero if <code>u &lt; v</code>; equal zero if <code>u = v</code>; and greater than zero if <code>u &gt; v</code>.</td>
<td><code>char u[]=&quot;ted&quot;;</code>&lt;br&gt;<code>char v[]=&quot;tom&quot;;</code>&lt;br&gt;<code>int c=strcmp(u,v);</code>&lt;br&gt;<code>cout &lt;&lt; c;</code>&lt;br&gt;Output -1</td>
</tr>
</tbody>
</table>

**Table 1.** Selected functions in library `string.h.  (#include <string>) continued**