Converts the sequence of multibyte characters pointed to by `s` into a sequence of wide characters, storing at most `n` wide characters in the array pointed to by `pwcs`. Conversion ends if a null character is encountered; it is converted into a null wide character.

Returns Number of array elements modified, not including the null wide character, if any. Returns `(size_t) (-1)` if an invalid multibyte character is encountered. *25.2*

`mbtowc` *Convert Multibyte Character to Wide Character* `<stdlib.h>`

```
int mbtowc(wchar_t * restrict pwc,
           const char * restrict s, size_t n);
```

If `s` isn't a null pointer, converts the multibyte character pointed to by `s` into a wide character; at most `n` bytes will be examined. If the multibyte character is valid and `pwc` isn't a null pointer, stores the value of the wide character in the object pointed to by `pwc`.

Returns If `s` is a null pointer, returns a nonzero or zero value, depending on whether or not multibyte characters have state-dependent encodings. If `s` points to a null character, returns zero. Otherwise, returns the number of bytes in the multibyte character pointed to by `s`; returns –1 if the next `n` or fewer bytes don't form a valid multibyte character. *25.2*

`memchr` *Search Memory Block for Character* `<string.h>`

```
void *memchr(const void *s, int c, size_t n);
```

Returns A pointer to the first occurrence of the character `c` among the first `n` characters of the object pointed to by `s`. Returns a null pointer if `c` isn't found. *23.6*

`memcmp` *Compare Memory Blocks* `<string.h>`

```
int memcmp(const void *s1, const void *s2, size_t n);
```

Returns A negative, zero, or positive integer, depending on whether the first `n` characters of the object pointed to by `s1` are less than, equal to, or greater than the first `n` characters of the object pointed to by `s2`. *23.6*

`memcpy` *Copy Memory Block* `<string.h>`

```
void *memcpy(void * restrict s1,
             const void * restrict s2, size_t n);
```

Copies `n` characters from the object pointed to by `s2` into the object pointed to by `s1`. The behavior is undefined if the objects overlap.

Returns `s1` (a pointer to the destination). *23.6*

`memmove` *Copy Memory Block* `<string.h>`

```
void *memmove(void *s1, const void *s2, size_t n);
```

Copies `n` characters from the object pointed to by `s2` into the object pointed to by `s1`. Will work properly if the objects overlap.

Returns `s1` (a pointer to the destination). *23.6*