```
void srand(unsigned int seed)
{
      next = seed;
}
```

7.20.3 Memory management functions

1 The order and contiguity of storage allocated by successive calls to the **calloc**, **malloc**, and **realloc** functions is unspecified. The pointer returned if the allocation succeeds is suitably aligned so that it may be assigned to a pointer to any type of object and then used to access such an object or an array of such objects in the space allocated (until the space is explicitly deallocated). The lifetime of an allocated object extends from the allocation until the deallocation. Each such allocation shall yield a pointer to an object disjoint from any other object. The pointer returned points to the start (lowest byte address) of the allocated space. If the space cannot be allocated, a null pointer is returned. If the size of the space requested is zero, the behavior is implementation-defined: either a null pointer is returned, or the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object.

7.20.3.1 The `calloc` function

Synopsis

1
```
#include <stdlib.h>
void *calloc(size_t nmemb, size_t size);
```

Description

2 The **calloc** function allocates space for an array of **nmemb** objects, each of whose size is **size**. The space is initialized to all bits zero.[261)]

Returns

3 The **calloc** function returns either a null pointer or a pointer to the allocated space.

7.20.3.2 The `free` function

Synopsis

1
```
#include <stdlib.h>
void free(void *ptr);
```

Description

2 The **free** function causes the space pointed to by **ptr** to be deallocated, that is, made available for further allocation. If **ptr** is a null pointer, no action occurs. Otherwise, if the argument does not match a pointer earlier returned by the **calloc**, **malloc**, or

261) Note that this need not be the same as the representation of floating-point zero or a null pointer constant.

realloc function, or if the space has been deallocated by a call to **free** or **realloc**, the behavior is undefined.

Returns

3 The **free** function returns no value.

7.20.3.3 The malloc function

Synopsis

1
```
#include <stdlib.h>
void *malloc(size_t size);
```

Description

2 The **malloc** function allocates space for an object whose size is specified by **size** and whose value is indeterminate.

Returns

3 The **malloc** function returns either a null pointer or a pointer to the allocated space.

7.20.3.4 The realloc function

Synopsis

1
```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

Description

2 The **realloc** function deallocates the old object pointed to by **ptr** and returns a pointer to a new object that has the size specified by **size**. The contents of the new object shall be the same as that of the old object prior to deallocation, up to the lesser of the new and old sizes. Any bytes in the new object beyond the size of the old object have indeterminate values.

3 If **ptr** is a null pointer, the **realloc** function behaves like the **malloc** function for the specified size. Otherwise, if **ptr** does not match a pointer earlier returned by the **calloc**, **malloc**, or **realloc** function, or if the space has been deallocated by a call to the **free** or **realloc** function, the behavior is undefined. If memory for the new object cannot be allocated, the old object is not deallocated and its value is unchanged.

Returns

4 The **realloc** function returns a pointer to the new object (which may have the same value as a pointer to the old object), or a null pointer if the new object could not be allocated.