CSE 374 Exam 3 Reference Sheet

C Library Header – stdlib.h

```
EXIT_SUCCESS // success termination code
EXIT_FAILURE // failure termination code

void* malloc (size_t size); // allocate size of memory block
void* realloc (void* ptr, size_t size); // change size of mem  block at ptr
void free (void* ptr); // free mem block at ptr, does nothing when ptr = NULL
void exit (int status); // terminate calling process
```

C Library Header – string.h

```
size_t strlen(const char* s); // calculates the length of s, excluding the '\0'
int isalpha(char c); // checks if c is an English alphabet character i.e. [a-zA-Z]
char* strcpy(char* dst, const char* src);
// strcpy copies src into dst, including the null terminator
char* strncpy(char* dest, const char* src, size_t n);
// strncpy behaves like strcpy(), but only writes at maximum n chars
char* strcat(char* dst, const char* src);
// strcat concatenates src onto the end of dst and adds a null terminator
```

C++ Standard Template Library (STL) – vector, list, map, etc.

```
.begin() // get iterator to beginning (first element)
.end() // get iterator to end (one past last element)
.size() // get container size
.erase(...) // erase element (pass 1 iterator) or range (pass 2 iterators)

template class std::vector;
     • .operator[](), .push_back(), .pop_back()
template class std::list;
     • .push_back(), .pop_back(), .push_front(), .pop_front()
template class std::map;
     • .operator[](), .insert(), .find(), .count()
```

C++ Smart Pointers Library – memory

`template class unique_ptr;`

- `.get()` // *get the raw pointer*
- `.reset()` // *stop managing the raw pointer and returns it*
- `.release()` // *release the ownership to another smart_ptr*

`template class shared_ptr;`

- `.get()`
- `.use_count()` // *returns ref count*
- `.unique()` // *returns whether the pointer is unique*

`template class weak_ptr;`

- `.lock(), .use_count(), .expired()`

On a 64-bit operating system, the following types have the following sizes

`char`	1	`size_t`	8
`short`	2	`long`	8
`int`	4	`void*`	8