CS111 Exam Reference Sheet

Filesystem Access
```
// specify one of O_RDONLY, O_WRONLY, O_RDWR in flags
// O_TRUNC clears existing file, O_CREAT creates if doesn't exist,
// O_EXCL fails if already exists
int open(const char *pathname, int flags); // returns descriptor
int open(const char *pathname, int flags, mode_t mode); // also sets permissions


int close(int fd); // ignore retval
int dup2(int oldfd, int newfd); // ignore retval
int pipe(int fds[]); // ignore retval
int pipe2(int fds[], int flags); // ignore retval, flags typically O_CLOEXEC
ssize_t read(int fd, void *buf, size_t count); // returns bytes read into buf
ssize_t write(int fd, const void *buf, size_t count); // returns bytes written

#define STDIN_FILENO 0
#define STDOUT_FILENO 1
#define STDERR_FILENO 2
```

Multiprocessing
```
pid_t fork();
pid_t waitpid(pid_t pid, int *status, int flags);
int execvp(const char *path, char *argv[]); // ignore retval
#define WIFEXITED(status) // macro
#define WEXITSTATUS(status) // macro
```

Multithreading
```
class thread {
public:
    thread(...); // first argument is function, its args come afterwards
    void join();
};
```

Use **ref()** in the thread constructor to pass args by reference

```
class mutex {
public:
    mutex();
    void lock();
    void unlock();
};

class condition_variable_any {
public:
    void wait(mutex& m);
```

```cpp
    void notify_one();
    void notify_all();
};
```

C++ Standard Library

```cpp
template <typename T>
class vector {
public:
    size_t size() const;
    void push_back(const T& elem);
    T& operator[](size_t i);
};

template <typename Key, typename Value>
class map {
public:
    size_t size() const;
    Value& operator[](const Key& key);
};
```