1 How do you use unnamed pipe to send a message from the parent to the child?
2. What is fseek and ftell? How would you use them?
3. What happens to the other process if you fclose after forking?
4. What happens to the other process if you fseek before forking?
5. What happens to the other process if you fseek after forking?
of what happens to the other process if you iscent after forming.
6. Why does pwrite exist? When would you use it?

- 7. What is an named pipe and an unnamed pipe?
- 8. What signals can a pipe generate and when?
- 9. How would you modify your pipe code to send an integer value of a variable?
- 10. Why is it necessary to close the pipe's unused filedescriptors after forking?

```
11. How would you fix/improve this code?
pthread mutex t m;
                                                  void* dequeue() {
pthread_cond_t cv;
                                                   p_m_lock(&m);
int in, out, count;
                                                   while(count == 0) {}
                                                   void* result = buffer[ (out++) % 16 ];
void* buffer[16]
                                                   p cond broadcast(&cv);
void enqueue(void* ptr) {
                                                   pthread_mutex_unlock(&m);
 p_m_lock(&m);
                                                   count --:
 while(count < 16) {}</pre>
                                                   return result;
 pthread_mutex_unlock(&m);
                                                  }
 p_cond_broadcast(&cv);
 count ++;
buffer[(in++) % 16] = ptr;
}
void pipe or quit(int*result) {
                                                  int run(const char *test, const char *prog, const char
if( 0 == pipe(result) ) return; else quit("pipe");
                                                  **args, const char *input, char **output,
                                                         char **erroroutput, int *waitresult) {
                                                   if (test) printf("%s: Running %s\n", test, prog);
                                                   int pipes[6];
void create_pipes(int* array6) {
pipe_or_quit(array6);
                                                   create_pipes(pipes);
pipe_or_quit(array6 +2);
                                                   pid_t childid = fork_or_quit();
pipe_or_quit(array6 +4);
                                                   if(childid == 0) {
                                                  //Child should close 'in'(input), out(output) err(output)
                                                    // close unused end of pipes
                                                    close(pipes[1]); close(pipes[2]); close(pipes[4]);
                                                    int old err fd = dup(2):
void exec_or_quit(const char *program, const
                                                    dup2_or_quit(pipes[0] /*read from */,0);
char **args, int old_err_fd) {
 execv(program, (char*const*) args);
                                                    dup2_or_quit(pipes[3] /*write to*/, 1);
                                                    dup2_or_quit(pipes[5] /*write to*/ ,2);
 dup2(old_err_fd, 2);
quit("execv");
                                                    alarm(ALARM_TIMEOUT_SECONDS);
                                                    exec_or_quit(prog, args, old_err_fd);
```