# **CSSE2310: 2014 exam answers**

## [**UQAttic**](http://uqattic.net)

## **Get more out of your study time. Join UQAttic's revision chat.**

#### [**Other exam papers**](https://drive.google.com/drive/folders/1k04K5RHL61k3ae2jJ362MPdN5xy3Bqsp)

### Please **contribute** to these documents.

### If you're looking for an effective way to familiarise yourself with the course material, you can't go past collaborating with fellow students. We have laboured to put these up, and so at the very least point out where you think we are wrong!

### You'll get more out of the course, you'll do better in the exam, and other students will benefit from your input as well.

### To get editing permissions, simply go to the [chatroom](http://uqattic.net) and provide us with your Google Account address.

### **Style.**

### Type answers in blue beneath each question.

### If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.

### If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

### **Communicate.**

### Head over to [uqattic.net](http://uqattic.net/) and click "Chat Now!". You'll find a chatroom full of students just like you. Talk about a revision document (like this one) or swap prep tips. If you have your own IRC client, point it to irc.uqattic.net, port 6667, channel #attic.

## 

Question 1)

1. Tail -2 strax  
   Or

Cat strax | tail -2

1. head 17 strax | tail -1  
   Or  
   Cat strax | head -17 | tail -1
2. grep “Clara \> River” file1
3. rm s\*?d

rm `ls s\*d | grep -wv sd` [JB]

1. grep “dinosaur” file1 > london

Or: grep -w dinosaur file1 >> london

Question 2)

1. char foo;
2. struct foo {  
    Int danny;  
    \_Bool pink[5]; // same as bool in c99  
   };

OR:

typedef struct {

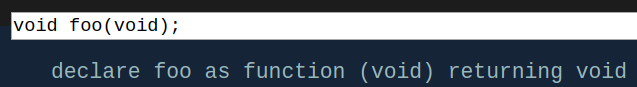
int danny;

bool pink[5];

} foo; [JB]

1. ~~void foo(){};  
   Shouldn’t this be   
   void foo(); ?~~

void foo(void);



1. char \*foo[17];
2. void (\*foo)(void);  
   ~~Could this also be  
   void (\*foo)(); ?~~

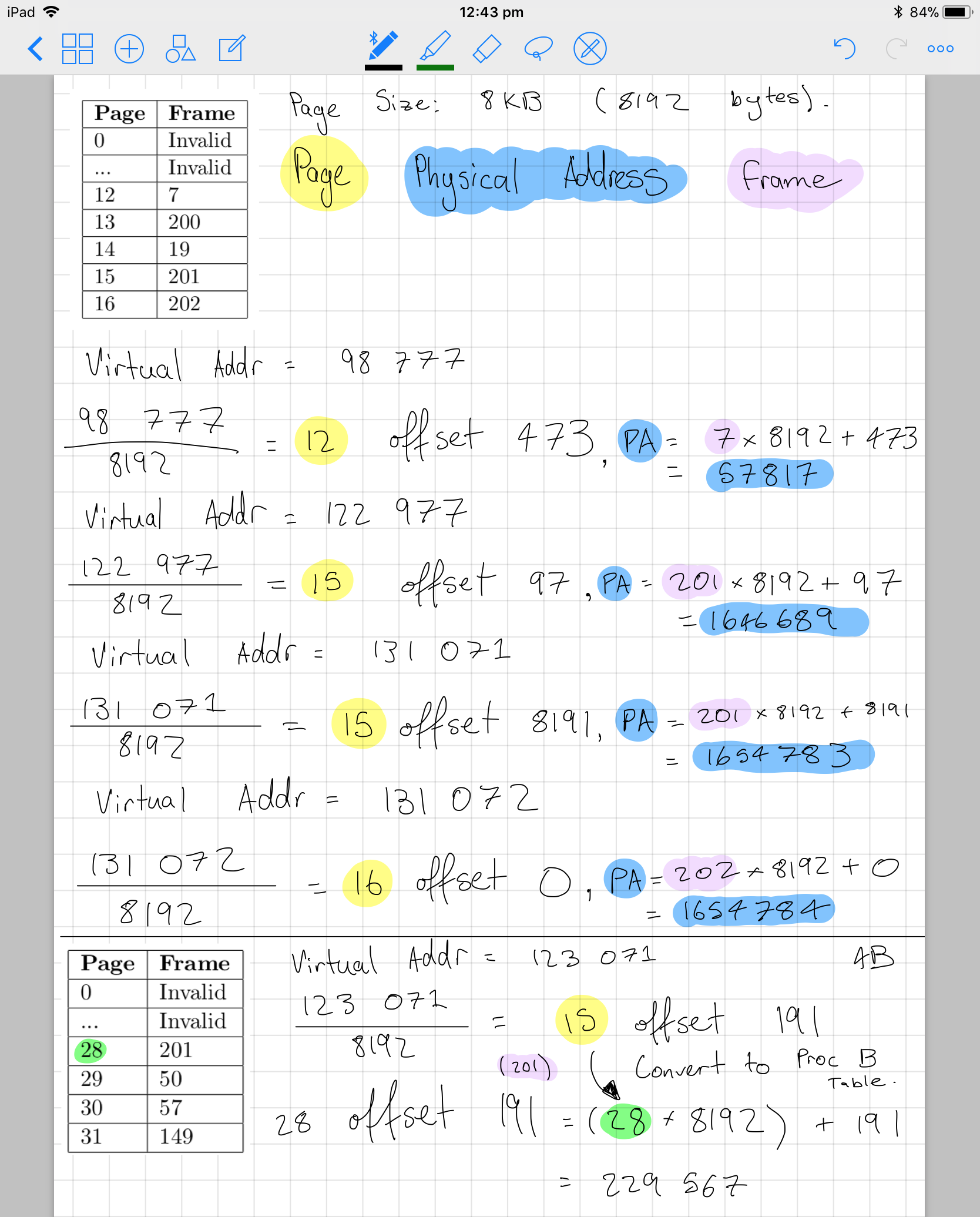
() and (void) do not mean the same thing

Question 3)

1. 13
2. 20
3. 1 12 should be 1 11
4. 11
5. 6 4
6. -2147483648 10 (?? this seems like a weird question for an exam)
7. 200
8. 24 9
9. 9.000000
10. 17.000000

Question 4)

1. i) 57817  
   ii) 1646689  
   iii) 1654783  
   iv) 1654784
2. 229567



1. The frame is invalid

Memory may not have been allocated to the process yet (this is within the frame).

Another answer:

Write on a read only page, or read on a write only page

Question 5)

1. Creating link with same name in the current directory

In the 2nd form, ln creates a link to TARGET in the current directory.

An invalid cross device link - i.e. The target and destination are on different partitions/drives. Alternatively, the file could already exist.

Hard links are not allowed for directories

1. 241772 drwx------ 5 uqjfenw1 uusers 56 Sep 16 12:14 ass3

The third column shows the number of hard links which for a directory is the number of subdirectories plus itself and its parent.

3

1. Hard links point to the memory location, not the file.

So nothing.

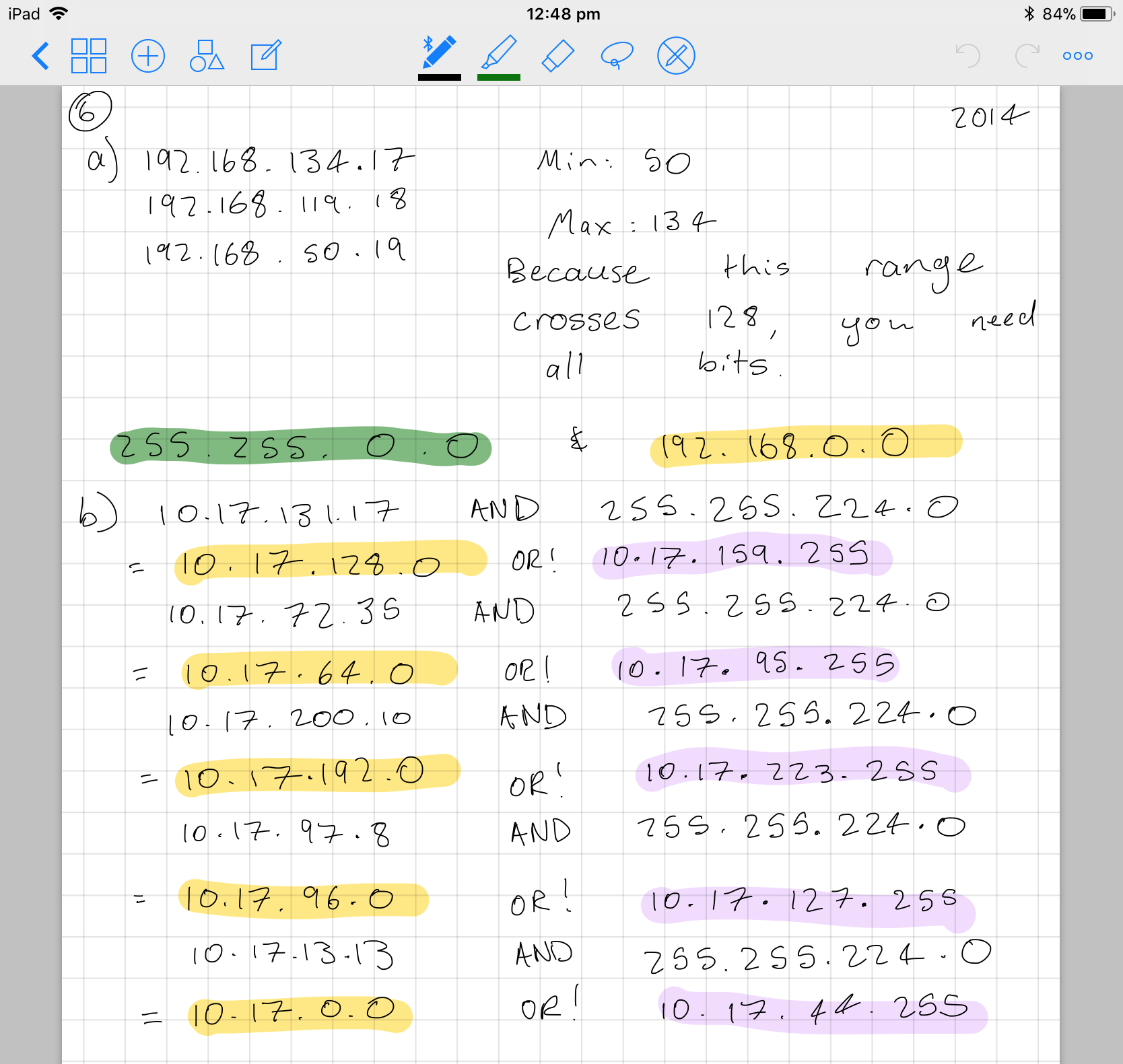
1. Export PATH = $PATH:/tmp/jfiles/grum.py
2. Chmod o+rx smoketest;  
   Chmod 11To1 smoketest;

chmod a+x smoketest; [JB] (+1)

1. It adds a link to an external piece of hardware. Adds an external folder.

Mounting enables access to a filesystem that was previously inaccessible. This could be a separate drive partition (sidenote: but it doesn’t have to be - e.g. ramfs). It grafts a subtree on to a folder within the tree of the root filesystem. [JB]

Question 6)



1. 192.168.0.0  
   255.255.0.0-
3. 10.17.128.0 | 10.17.159.255  
   10.17.64.0 | 10.17.95.255  
   10.17.192.0 | 10.17.223.255  
   10.17.96.0 | 10.17.127.255  
   ~~10.17.0.0 | 10.17.44.255~~

10.17.0.0 | 10.17.31.255 +4

1. Because data can be stored in different endian order to what is required (networks are big endian).
2. Waits for an incoming connection and gets a file descriptor for it.
3. i) Physical  
   ii) physical, link  
   iii) application  
   iv) network
4. Transport  
   Application  
   Network  
   Application  
   Transport  
   Application

Question 7)

1. 100679760kB = ~98320 MB = ~96 GB

I agree.

8 KiB \* (10 + 2048 + 3\* 2048^2) = 100679760 KiB = ~96.016 GiB [JB]

I think ^ used 8096 not 8192 for 8KB   
8192\*(10+(8192/4)+3\*(8192/4)^2) = ~103.1 GB = ~96.016 GiB

You made the exact mistake you are talking about but AFTER the calculation. You forgot to divide your final solution by 1024 rather than 1000. This is why I’m specific by using the KiB/GiB units.

1. 6

1+1+1+2+1 = 6

1. 2058

Question 8)

1. i) both  
   ii) threads  
   iii) threads +1

Is this both? Because for process it’s waiting on its grandchild.

Seconding both. Process can be reaped by init or its parent only.  
 iv) both

1. Waitpid and sigchild handler  
   Since there’s no mention of a specific child, I wrote “wait” as well as “waitpid”
2. WIFEXIT & WEXITSTATUS
3. SIGKILL #9
4. sigaction
5. 5



I disagree:

|  |  |  |  |
| --- | --- | --- | --- |
| \_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | | Parent pid!=0 | | | |**\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | | | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | | |  | |**\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | Child pid==0 |  | |**\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

[JB]

1. i) will return 0; **will not return 0, will actually terminate with sigpipe upon entering the second loop.**  
   ii) 5; Tested and found to be correct

Question 9)

// This is slightly broken when printing the end column of a cut

bool chopLines(int n, char c, const char\* filename) {

int i, start;

char line[81];

FILE\* myFile;

myFile = fopen(filename, "r");

if (myFile == NULL) {

return false;

}

while((fgets(line, 81, myFile))) {

start = 1;

for (i = 0; i < 81; i++) {

if(line[i] == '\0') {

break;

}

if(start == n) {

while(line[i] != c) {

printf("%c", line[i++]);

}

printf("\n");

}

if(line[i] == c) {

start++;

}

}

}

return true;

}

// Intended to execute the cut program described, but works with the regular posix cut by adding “-f” and “-d” to the arguments

bool cutChop(int n, char c, const char\* filename) {

int pid = fork();

if (pid == 0) { // child process

char num[3];

char letter[2];

sprintf(num, "%d", n);

letter[0] = c;

letter[1] = '\0';

execlp("cut", "cut", num, letter, filename, NULL);

} else { // parent process

int status;

waitpid(pid, &status, 0);

return WIFEXITED(status == 7);

}

return true;

}



struct Args {

int n;

char c;

char\* filename;

};

void\* start\_thread(void \*vargp) {

struct Args \*args = (struct Args\*)vargp;

bool retVal = cutChop(args->n, args->c, args->filename);

return &retVal;

}

bool chopLinesMany(int n, char c, const char \*\*filenames, int numFiles) {

struct Args args;

args.n = n;

args.c = c;

pthread\_t\* threads = malloc(numFiles \* sizeof(pthread\_t));

for (int i = 0; i < numFiles; i++) {

pthread\_t threadID;

/\* Loop through files\*/

args.filename = filenames[i];

pthread\_create(&threadID, 0, start\_thread, (void\*)&args);

threads[i] = threadID;

}

/\* Wait for all threads to end \*/

bool retVal, myRet = true;

for (int i = 0; i < numFiles; i++) {

pthread\_join(threads[i], (void\*\*)&retVal);

if (retVal == false) {

myRet = false;

}

}

return myRet;

}

d)

bool cutChopMany(int n, char c, const char\*\* filenames, int numfiles) {

pid\_t processID;

pid\_t\* ids = malloc(numfiles \* sizeof(pid\_t));

for (int i = 0; i < numfiles; i++) {

processID = fork();

if (processID == 0) { // child process

return cutChop(n, c, filenames[i]);

} else { // parent process

ids[i] = processID;

}

}

/\* Wait for all of the return values \*/

int\* retVals = malloc(numfiles \* sizeof(bool));

bool myRet = true;

for (int i = 0; i < numfiles; i++) {

waitpid(ids[i], &(retVals[i]), 0);

if (retVals[i] == false) {

myRet = false;

}

}

return myRet;

}