# **CSSE2310: 2012 exam answers**

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

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

#### [**Other exam papers**](https://docs.google.com/folder/d/0B6_D4T6LJ-uwYzY1YWMzNjYtMzUyZC00OTEyLWJlMjktOGExYWUwOTc4NDE3/edit)

### 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: Shell Stuff

A) Rename file f1 to f2.

mv f1 f2

B)Copy all files (in the current directory) which have exactly six characters in their names to /tmp.

~~cp `ls | grep ??????` tmp~~ missing the / in front of the tmp makes this wrong, the ls fancyness isn’t required

cp ?????? /tmp <- ?????? means all files with 6 chars [+2]

cp . /tmp (since . refers to current directory?)

C)Show the 42nd line of words.txt.

~~more +42 words.txt | head -n 1~~ this is equivalent to just head -n1 words.txt, thus it is wrong

I don’t understand; Why is it equivalent to head -n1? It works on moss?

It’s not, they’re talking smack

head -42 words.txt | tail -1 ← thats how i did it and it works [+2]

Alternatively, you can use `sed ‘42q;d’ words.txt` or `awk ‘NR==42’ words.txt`

I prefer `sed -n 42p words.txt` [+1]

D)Find all the lines in doc.txt which contain the word “fez” and store them in a file called cool.txt

grep fez doc.txt > cool.txt [+1]

E) A file called words.txt has one word per line. Show how many different words in the file start

with “A”.

grep ^A words.txt | uniq | wc -l

Doesn't it need to sort before uniq since uniq only compares adjacent lines?

grep ^A words.txt | wc -l ←- this count repeated words

or

grep ^A -c words.txt ←- this count repeated words

(Different red here:) I think reading the question as “how many unique words” rather than “how many instances of words” is valid but unnecessary.

Question 2: C declarations

A) A long integer.

long int foo; (This is valid but there’s no need for ‘int’ here)

long foo;

B) An array of 42 strings.

char\* foo[42]; [+1]

C) A struct type containing an array of four ints and a char.

struct foo {

int fooInt[4];

char fooChar;

};

Note also the lowercase ‘foo’, as the question asks for ‘foo’, not ‘Foo’. (The style guide does not apply for the question, when it specifically asks for ‘foo’)

This question is somewhat ambiguous, as there is no such thing as a ‘struct type’ in the C grammar. There are type identifiers and struct identifiers. Due to this, one interpretation of the question is that it is asking for a typedef struct, as below. If this is on an exam, seek clarification as to whether they are looking for a type (typedef) or just a struct declaration.

typedef struct {

int fooInt[4];

char fooChar;

} foo;

Different red here: Agreed. It’s worth noting that “foo” in the blue solution is the name of a tag, not a type. You would still need to declare a variable of that type as follows:

struct foo bar;  
Where bar is the actual variable.

D)A pointer to the struct described in part C.

struct foo\* foo; [+1]

(Yes, this is legal C, with both being lowercase)

Following from the previous answer, if the interpretation

of 2C is asking for a typedef, then you would simply have:

foo \*foo;

E)Another name for int.

typedef int foo; [+1]

Question 3: C code output

A)

5

B)

1

C)

5

D)

3 4

E)

14 4

F)

21

G)

~~5~~

5.000 [confirmed]

Should be 5.00 because it’s float.

H)

Ef

Should this be def, since in the if statement, they set d=true?

No, answer is ‘ef’, reason is with the if statement, as soon as the first ‘true’ statement is reached with a logical or, the function continues and ignores (d=true).

I)

1004

// This makes sense but I somehow get 1016 when I run it on moss (also with a few casting warnings)...

^Moss uses a different sized integer to the one acknowledged in this question

Wouldn’t it be because, when p is increased, it’s increased by 4, rather than 1 (since pointers aren’t incremented by 1 address, but by the size of what they’re pointing to)?

So:

p+sizeof(int)=p+4=1000+4(int) = 1000 + 16 = 1016

See: <http://stackoverflow.com/q/26908009/604114> for a detailed discussion.

J)

3

https://en.wikipedia.org/wiki/Dangling\_else#C

Question 4: circular list find code

This would not work as it’s not an array.

cnode\* find(clist\* cl, int v) {

struct cnode firstNode = clist[0];

struct cnode curNode = firstNode;

int i = 0;

while(curNode.data != v) {

if((&curNode == &firstNode) && (i !=0)) {

return NULL;

}

i++;

curNode = cList[i];

}

return &curNode;

}

cnode\* (clist\* cl, int v) {

if (cl->head->data == v) { // Check the head's data

return cl->head;

}

cnode\* first = cl->head->next; // The first element in the list

cnode\* current = cl->head; // Whatever we are currently checking

while ((current = current->next) != first) {

if (current->data == v) {

return current;

}

}

return NULL;

}

cnode\* find(clist\* cl, int v) {

if (cl->head == NULL) {

return NULL;

}

struct cnode\* firstNode = cl->head->next;

struct cnode\* currNode = cl->head->next;

do {

if (currNode->data == v) {

return currNode;

}

} while (curr->next != firstNode && currNode->currNode->next);

return NULL:

}

^ Would that be right?

meant to be && currNode = currNode->next?

Question 5: Memory stuff

A)

page size = 2 \* 1024^2 = 2097152 bytes

Page# = floor(virtual address / page size)

5,5,5,6,10

(The third one is 5 because of the green explanation below).

Very likely incorrect:

5, 6, 6, 6, 10

*I get the same thing except 5 for the 2nd one, but I have no idea if I’m doing it correctly. I said each page is 2MB so the page number is whatever’s left after removing up to the 21st (as 2MB = 221 bits) binary digit.*

*^ I agree*

*I also agree, except the tutor Simon said if the address divide 2^21 is exactly a number, the page it’s in is the page above it. Didn’t really understand why though. Making it 5, 5, 7, 6, 10.*

*^ well if you think about it, the page is going to span for lets say 100 hundred “lines”, which is from 0 - 99, so if you get something that is a whole number that is not 0, you’re on line “100” which is on the next page? Something like that, hope this explanation doesn’t confuse anyone further :X*

*^^ Third one can’t possibly be 7, since 1258912 < 1258929. If pages are indexed starting from 0, and they have 10 bytes, then address 10 will be on the 2nd page, i.e. page 1, so the formula floor(address/page size) holds perfectly for 0 indexed pages, hence the answer is 5, 5, 6, 6, 10 [BenM]*

B)

also likely incorrect:

line 1

*~~I said line 10 since segfaults are only caused when you try to write to the address (It’ll probably show some garbage if you just read), and also e is a null pointer~~*

*~~nvm, just compiled it and it actually died at line 0, but after commenting that out it died at line 1.~~*

On second thought, since “\*a=0” runs, you can assume it’s in page 5 so line 1-3 should be fine but could segfault at 4. (maybe?)

According to Simon, the tutor: “b cannot segfault since it’s on the same page as a [page 5]”.

So our program has access to page 5, but c is on page 6, which “may be invalid or you don’t have appropriate permissions”, so c is the answer - the first line which may segfault.

So should this be line 4 which causes segfault or line 5?????

C)

Segfaults occur when either (1) an invalid page is accessed or (2) one tries to write to a read-only page.

In the for loop, one is only ever reading from the page in which ‘f’ lives. The segfault could possibly occur if that page is read only (hence you would be writing to a read only page).

Question 6: i-node file system stuff

A)

i)

Assuming counting of blocks starts at 0, 1037

ii)

assume table size of 1024

Not assumed, can be calculated: blocks are 4KB, each pointer is 4 bytes so 2^10 block pointers per index file gives 1024.

Blocks 0 to 12 are pointed to by the first 13 direct pointers.

Blocks 13 to 1037 are referenced by the first pointer in the first block of the indirect pointers.

Should it be 13 to 1036? Because 12+1024=1036.

To read 14, we need 2 reads (one for the pointer and one for the data). Since we have already cached the pointers used for 1026 and 1037, we only require 1 additional read for each. Total of 4 reads.

B)

All units in bytes.

a = number blocks mapped by direct blocks (13 in this question)

b = number blocks mapped by indirect blocks (1 in this question) \* block\_size

c = number blocks mapped by double indirect blocks (1 in this question) \* block\_size^2

d = number blocks mapped by triple indirect blocks (0 in this question) \* block\_size^3

Max File size = block\_size \* (a + b + c + d)

i)

Change is:

[(12\*4096) + (2\*4096\*1024) + (1\*4096\*1024^2)] -[(13\*4096) + (1\*4096\*1024) + (1\*4096 \* 1024^2) ] = 4,190,208 Bytes

Could be factorised to:

[4096 \* (13 + 1024 + 1024^2)] - [4096 \* (12 + 2\*1024 + 1024^2)] = 4,190,208 Bytes

^^ Even more simply, change in max file size = (increase in block pointers - decrease in block pointers) \* block size:  
delta = (1024 - 1) \* 4KB = 4092KB [BM]  
  
Inceasing 1 single indirect would be = 4096 \* 1024 \* 1 = 4194304  
Hence the total increase in the file size= 1 single indirect - direct  
 = 4194304 - 4096  
 = 4190208

ii)

Change is:

[(13\*4096) + (1\*4096\*1024) + (1\*4096 \* 1024^2) ] -

[(13\*4096) + (0\*4096\*1024) + (2\*4096 \* 1024^2) ] = 4,290,772,992 Bytes

^^ As for i:  
delta = (1024^2 - 1024) \* 4KB = 1024 \* (1024 - 1) = 4092MB [BM]  
  
Change is = (1024^2 \* 4096) - (1024\*4096) = 4290772992

C)

Data could be accessed with greater speed. Going through one indirect pointer only requires one additional read, going through a double indirect pointer requires 2 additional reads.

^^ Where is there mention of a double indirect pointer?

My answer was: It would allow faster access to the 14th block of the file since it would no longer require the level of indirection required by the single indirection pointer. [BM]

Would the 14th block still require the level of indirection required by the single indirection pointer? Because 14th block is still outside of the range [0, 13] mapped by direct blocks (it was [0, 12] before adding an additional direct pointer).

Question 7: File permission stuff

A)

write (not read or execute)

B)

Read and execute (not write)

C)

What is meant by “access”?

chmod 070 lib

chmod 770 lib

- Wouldn’t 770 make more sense since joel is also a member of staff?

Yes, we don’t want to overwrite the permission for Joel in this case.

D)

There is a hardlink between the two (both refer to the same file). Their permissions are not

consistent. This is impossible, since they are both names for the same file (which should have the same permissions).

E)

Because the directory /usr/local/things probably doesn’t permit execution of files for donny.

To expand on this - if donny doesn’t have +x rights on the directory /usr/local/things, he’s not allowed to go through it to access any of its contents, even if /usr/local/things/program has a+rwx.

Could it also be that program has been deleted and the softlink remains?

I think this is unlikely, since the permissions were given, and permissions don’t exist for non-extant files. [BM] +1

Bit confused on this, the permissions show that user, group and others can execute this file. Donny is user, shouldn’t he be able to execute this file?

Question 8: Fork stuff

A)

i)

b, c, d? D can only connect with c

Think it’s just b and c - d is a grandchild of a, not a child, so it cannot have a direct pipe connection. I agree with this. +1

Since each of the children inherit the same pipes, why couldn’t d utilise a pipe created by a (and passed on by c)?

Why can’t process ‘a’ connect to itself with a pipe?

ii)

a

B)

i)

Down arrows indicate reaping. The function to achieve this in C is wait().

ii)

It was the parent of at least one child, grandparent of at least 2 grandchildren, great grandparent of at least one child? It has no zombies at the end?

It must be init; only a parent can reap its child and process X is reaping grandchildren so it must be init.

It’s interesting to note that init spawns every other process on the system. That’s how the processes being reaped are great-great-etc-grandchildren and thus related in the diagram.

C)

I’m assuming this question means “when is WIFEXITED false?”. A process won’t have a proper return value if a signal ended the process, although it will return some numbers depending on the signal which caused it to terminate (like 139 for segfault).

Question 9: Network Stuff

A)

~~Network address: 12.16,1,248~~

~~Subnet Mask: 255.255.255.248~~

Subnet masks must be of the form 1111...0000 (may or may not include zeros but can’t have 11001 or anything mixed like that)

network = 12.16.0.0

mask = 255.255.254.0

forest interface 1 has address 12.16.**0**.7 so you can’t have less than 254 in that third octet

B)

Catflap is gateway (= router?) for entire network?

forest, savanna are gateways (= routers?) for subnets?

forrest doesn’t exist in network?

Catflip, forest and savanna are all routers for some network.

C)

12.16.0.7

D)

To send a UDP packet we need access to the DNS server? So must go through Savanna. Also need to go through the main gateway (Catflap) and subnet gateway (Forest).

E)

All of the IPs under House are non-routable. This means “the internet” won’t understand them. The House router has to translate the non-routable IPs to routable IPs. This is called a Network Address Translator (NAT).

F)

socket()

bind()

listen()

accept()

G)

|  |  |  |
| --- | --- | --- |
| Number | Name | Address |
| 5 | Application | Leave Blank |
| 4 | Transport | TCP/(port?) |
| 3 | Network | IP |
| 2 | Link | Mac |
| 1 | Physical | Leave Blank |

Question 10: More C programming

A)

char\*\* arrsort(int size, const char\* const arr[]) {  
 char\*\* array = malloc(size \* sizeof(char \*));  
 char\* temp = malloc(sizeof(char));  
   
 //First copy all strings in arr over to array  
 for(int k = 0; k < size; ++k) {  
 array[k] = malloc((strlen(arr[k])+1) \* sizeof(char));  
 strcpy(array[k], arr[k]);  
 }  
   
 // Now sort array  
 for(int i = 1; i < size; ++i) {  
 for(int j = 0; j < size - i; ++j) {  
 temp = realloc(temp, strlen(array[j]) \* sizeof(char));  
 // If the elements are backwards  
 if(strcmp(array[j], array[j+1]) < 0) {  
 // put element j in a temp string  
 temp = realloc(temp, strlen(array[j]) \* sizeof(char));  
 strcpy(temp, array[j]);  
 // put element j+1 in element j  
 array[j] = realloc(array[j], strlen(array[j+1])\*sizeof(char));  
 strcpy(array[j], array[j+1]);  
 // put the temp str into element j + 1  
 array[j+1] = realloc(array[j+1], strlen(temp) \* sizeof(char));  
 strcpy(array[j+1], temp);  
 }  
 }  
 }

return array;  
}

I don’t think you need to allocate memory, I just swapped the pointers around:

char \*\*arrsort(int size, const char \*const arr[]) {

char \*\*sorted = malloc(size \* sizeof(char \*));

char \*temp;

int i, j;

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

sorted[i] = (char \*)arr[i];

}

for (i = 1; i < size; i++) {

for (j = 0; j < (size - 1); j++) {

if (strcasecmp(sorted[j], sorted[j+1]) > 0) {

temp = sorted[j];

sorted[j] = sorted[j+1];

sorted[j+1] = temp;

}

}

}

return sorted;

}

^this above solution doesn’t actually use the out most for loop(i): perhaps a better version of this code would be the folllowing.(it also creates a copy of the array).

char \*\*arrsort(int size, const char \*const arr[]) {

char \*\*sorted = malloc(size \* sizeof(char \*));

char \*temp;

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

sorted[i] = malloc(strlen(arr[i]));

strcpy(sorted[i], arr[i]);

}

for (int i = 1; i < size; i++) {

for (int j = 0; j < (size - 1); j++) {

if (strcasecmp(sorted[j], sorted[i]) > 0) {

temp = sorted[i];

sorted[i] = sorted[j];

sorted[j] = temp;

}

}

}

return sorted;

}

Another soln:

char\*\* arrsort(int size, char\*\* array){

//swap

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

for (int j = i; j < size; j++){

if (strcmp(array[j], array[i]) < 0){

char \* temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

//copy

char \*\* copy = malloc(sizeof(char \*)\*size);

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

copy[i] = malloc(strlen(array[i])\*sizeof(char));

copy[i] = strcpy(copy[i], array[i]);

}

return copy;

}

:

B)

// It is highly likely that this is all wrong… Please feel free to correct it

#define READ 0

#define WRITE 1

char\*\* psort(int size, char\* arr[]) {

int fd1[2];

int fd2[2];

pipe(fd1);

pipe(fd2);

pid\_t pid;

if ((pid = fork())) { // In the parent process

FILE\* toSort = fdopen(fd1[WRITE], "w");

FILE\* fromSort = fdopen(fd2[READ], "r");

close(fd1[READ]);

close(fd2[WRITE]);

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

fprintf(toSort, "%s\n", arr[i]);

}

fflush(toSort);

close(fd1[WRITE]);

// Now to read back the sorted data

char\*\* result = malloc(size \* sizeof(char\*));

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

char buffer, \*word = malloc(sizeof(char));

int allocated = 1;

while ((buffer = fgetc(fromSort)) != '\n' && buffer != EOF) {

if (strlen(word) >= allocated - 1) {

allocated \*= 2;

word = realloc(word, allocated \* sizeof(char));

}

word[strlen(word)] = buffer;

}

result[i] = malloc(sizeof(char) \* strlen(word));

strcpy(result[i], word);

free(word);

}

close(fromSort);

return result;

} else { // Child

close(fd1[WRITE]);

close(fd2[READ]);

dup2(fd1[READ], STDIN\_FILENO);

dup2(fd2[WRITE], STDOUT\_FILENO);

close(fd1[READ]);

close(fd2[WRITE]);

execlp("/bin/usr/sort", “sort”, NULL);

}

}

Not sure why the above outputs garbage, but I rewrote it slightly and it works fine: [BM]

#define READ 0

#define WRITE 1

char\*\* psort(int size, char\* arr[]) {

int fd1[2];

int fd2[2];

pipe(fd1);

pipe(fd2);

pid\_t pid;

if ((pid = fork())) { // In the parent process

FILE\* toSort = fdopen(fd1[WRITE], "w");

FILE\* fromSort = fdopen(fd2[READ], "r");

close(fd1[READ]);

close(fd2[WRITE]);

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

fprintf(toSort, "%s\n", arr[i]);

}

fflush(toSort);

close(fd1[WRITE]);

char \*line = NULL;

size\_t len = 0;

ssize\_t read;

char \*\*result = malloc(size \* sizeof(char \*));

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

char buf;

int buf\_size = 1;

int word\_size = 0;

char \*word = malloc(buf\_size \* sizeof(char));

while((buf = fgetc(fromSort)) != '\n') {

if(word\_size >= buf\_size) {

buf\_size \*= 2;

word = realloc(word, buf\_size \* sizeof(char));

}

word[word\_size++] = buf;

}

result[i] = word;

}

return result;

} else { // Child

close(fd1[WRITE]);

close(fd2[READ]);

dup2(fd1[READ], STDIN\_FILENO);

dup2(fd2[WRITE], STDOUT\_FILENO);

close(fd1[READ]);

close(fd2[WRITE]);

//execlp("/bin/usr/sort", "sort", NULL);

execlp("sort", "sort", NULL);

}

return NULL;

}

C)

int main(int argc, char\* argv[]) {

// sort words and check for invalid input

if(argc < 3) {

fprintf(stderr, “Not enough words\n”);

exit(1);

}

if(strcmp(argv[1], “a”) == 0) {

char\*\* sorted = asort(argc - 2, &(argv[2]));

} else if (strcmp(argv[1], “p”) == 0) {

char\*\* sorted = psort(argc - 2, &(argv[2]));

} else {

fprintf(stderr, “Unknown method\n”);

exit(2);

}

// Now print and free

for(int i = 0; i < argc - 2; i++) {

fprintf(stdout, “%s\n”, sorted[i]);

free(sorted[i]);

}

free(sorted);

return 0;

}