# **CSSE2310: 2014 midsem 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.

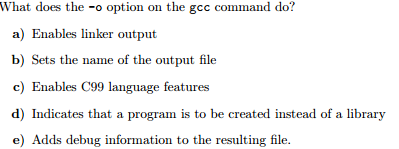
### If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

### 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.

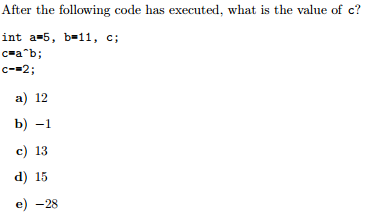
CSSE2310 2014 Mid-sem Answers

# Question 1



Answer: b) Sets the name of the output file

# Question 2

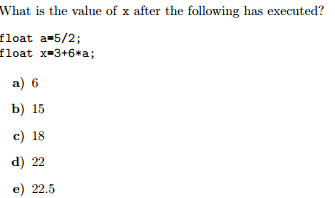


Convert ‘a’ and ‘b’ into binary: a = 0101, b = 1011

c = a XOR b = 1110 = 14  
14 - 2 = 12

Answer: a) 12

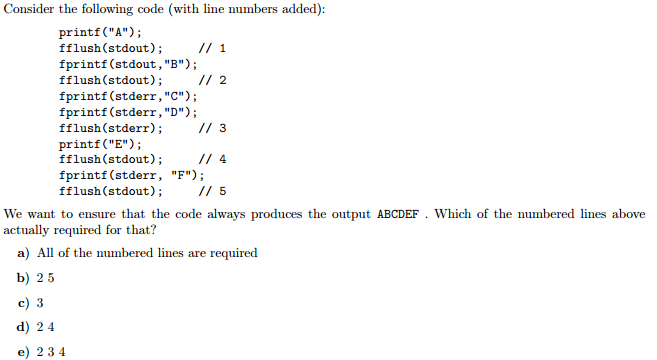
# Question 3



Float a = 5/2 = 2.0 (Integer division then cast to float)  
float x = 3 + 6\*2 = 15

Answer: B (X = 15)

# Question 4

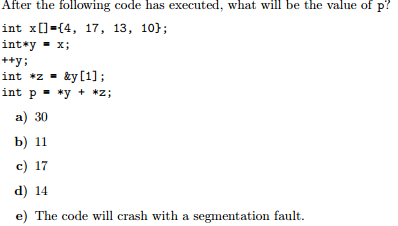


**Answer: D (2, 4)**

Stderr doesn’t buffer so output is immediate.   
1 Is not needed because 2 will flush both “A” and “B” in the correct order, A does not have to be flushed before we enqueue “B” in the buffer.  
2 is needed to flush “AB” before we output to stderr since stderr will print immediately.   
3 is not needed because stderr is unbuffered (it will print immediately).

4 is needed because we are about to use stderr  
5 is not needed for the same reason as 3

# Question 5



Let ‘->’ mean “points to.”

x = [4, 17, 13, 10]  
y -> x[0]   
Next we increment y:

y = y + 1 == &x[0] + 1 == &x[1]

Therefore y -> x[1]. For the purpose of making the next line more intuitive, this is the same as saying

&y[0] = &x[1] Which reads: “The address of the first element of y is assigned to be the address of the second element of x.”

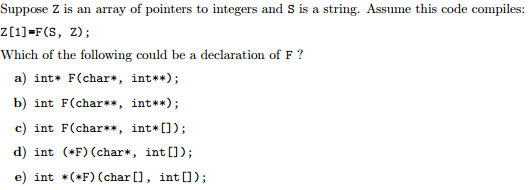
Then, z = &y[1] == y + 1 == x + 2 == &x[2]

So now we have, y -> x[1] and z -> x[2]

Therefore, \*y + \*z = x[1] + x[2] = 17 + 13 = 30.

Answer: a) 30

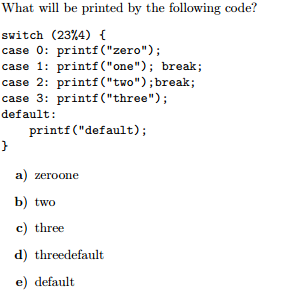
# Question 6



We can immediately dismiss d, e, because second param isn’t int\*\* or int\*[]  
Similarly, we can also dismiss b, c, because the first param isnt char\* or char[].  
Process of elimination leaves a).

Answer: a) int\* F(char\*, int \*\*);

# Question 7

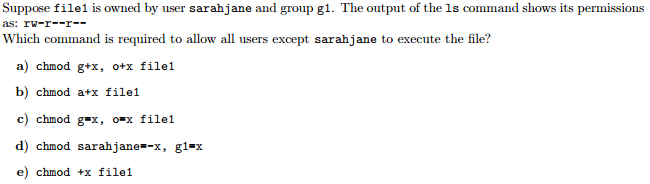


23 % 4 = 3  
This means case 3 runs, which prints “three”. However, there is no break at the end of case 3, and so we continue until we hit the next break, or the end of the switch statement. Effectively, this means that the default case is also executed, which prints “default”.

Answer: d) threedefault

(Although technically the program wouldn’t compile because default is missing a closing quote)

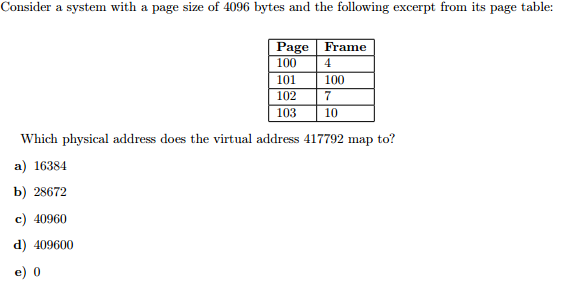
# Question 8



**Answer: A (chmod g+x, o+x file1)**

G+x means add executable to group, o+x means add executable to others  
This leaves u=rw- and g=r-x, o=r-x (IE: No execute bit for sarahjane)  
Note: u = user, g = group, o = others. Permissions are granted in that order (IE: if you are user and can’t execute then that overwrites your group based permissions, which may include execute)

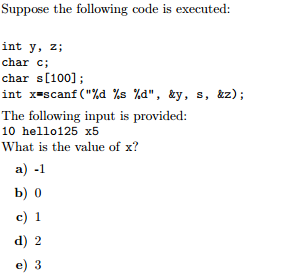
# Question 9



**Answer: B (28672)**

417792 / 4096 = 102, means on 102nd page. 102 page = frame 7, 7\*4096 = 28672

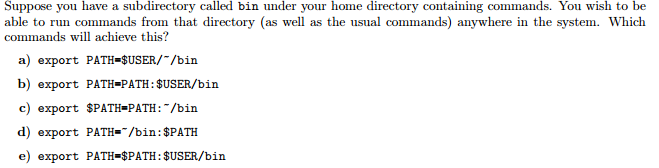
# Question 10



**Answer: D (2)**

The first two parameters are matched, whilst the last one fails to read as it is a string. Therefore scanf returns 2 (The number of successfully read arguments)

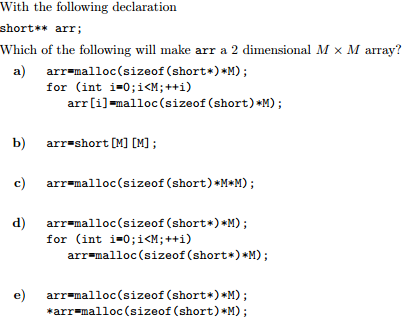
# Question 11



**Answer: D (export PATH=~/bin:$PATH)**

~/bin evaluates to home directory/bin and the :$PATH adds the pre-existing path variable to the end.

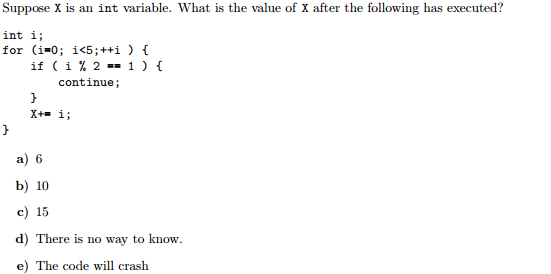
# Question 12



**Answer: A**

B assigns short[M][M] to arr, not legal because no type is specified  
C Allocates enough memory, but we won’t be able to access it like a 2 dimensional array  
D makes an array of arrays of short pointers (Not what we want)  
E allocates enough memory for the top level array and then points the first element to memory to hold the bottom level array, but doesn’t do that for the rest of the top level elements.

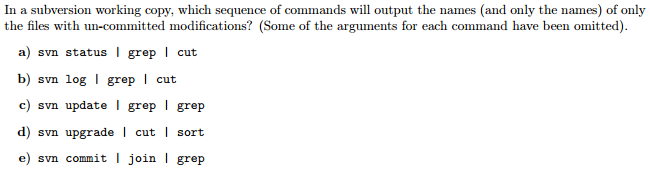
# Question 13



**Answer: D (There is no way to know)**

The code should work fine in ANSI C with –Wall and –pedantic. However, since we don’t know an initial value for X, we have no idea what it will evaluate to after we add to it.

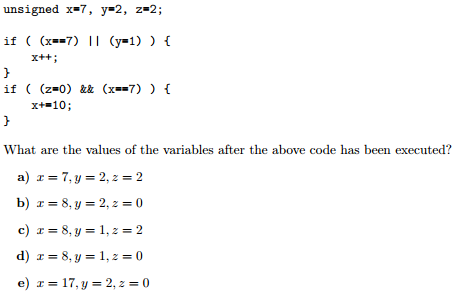
# Question 14



**Answer: A (svn status | grep | cut)**

Possible candidate: svn status | grep ? | cut –c 9-  
Grabs status, and only lines with modifications (?) and then returns everything after the 9th character (The filename itself)

# Question 15



In the first if statement, x == 7 evaluates to true, and so the OR operation short circuits. This means that if the first operand is true, then the logical OR is true, and therefore, c will not evaluate the second operand to save time. This might seem weird when there are two operands, but imagine the following { a() || b() || c() || … || z() } where each letter stands for a boolean function. This makes it easier to see why it’s more efficient to stop at the first true.

Anyway, the first if statement is true and so we increment x. For the second if statement, z = 0 assigns 0 to z and then evaluates to 0, or false. Since false and anything evaluates to false, c will also short circuit here, and in either case, x+=10 is not evaluated.  
This gives, x = 8, y = 2, z = 0.

**Answer: B (x=8, y=2, z=0)**