COSC 310 Final Review

The final exam is closed book and closed notes. You are allowed to bring one page of note sheet or reference card. However, the note sheet should not contain any pre-worked problems. I will provide scratch paper attached at the end of the exam paper.

You should read the textbook carefully, especially the main chapters listed below. To study, work the practice problems and review the homework questions and problems from midterm. Below, I also listed some sample questions (with solutions).

Main Chapters

- Chapter 1-3, focus on 3.6-3.8, 3.12, 3.13

  Condition codes, loop, procedure, array, buffer overflow

- C programming

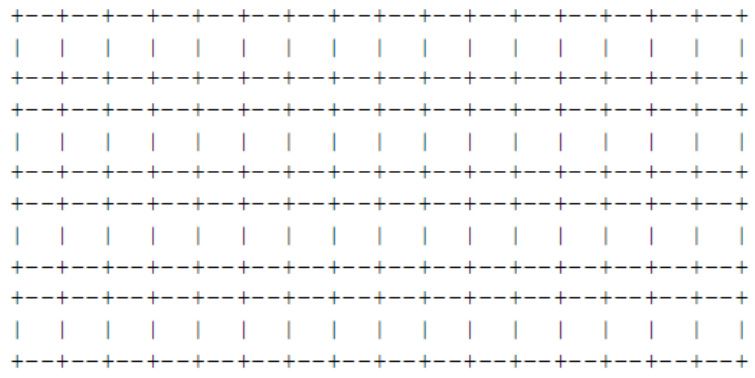
Final Review Sample Questions

Struct alignment

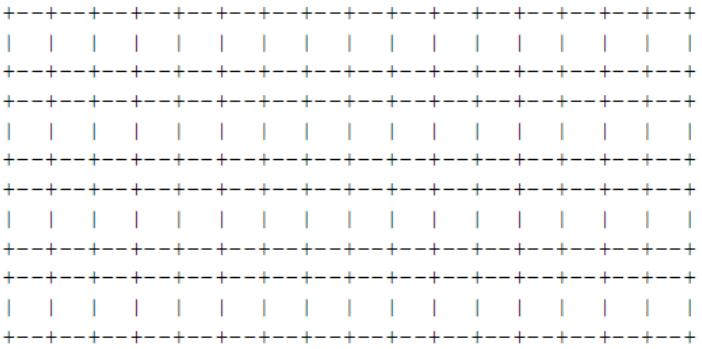
Consider the following C struct declaration:

```
typedef struct {
  char a;
  long b;
  float c;
  char d[3];
  int *e;
  short *f;
} foo;
```

1. Show how `foo` would be allocated in memory on an x86-64 Linux system. Label the bytes with the names of the various fields and clearly mark the end of the struct. Use an X to denote space that is allocated in the struct as padding.



2. Rearrange the elements of `foo` to conserve the most space in memory. Label the bytes with the names of the various fields and clearly mark the end of the struct. Use an X to denote space that is allocated in the struct as padding.



Solution:

```
1.
a X X X X X X b b b b b b b b
c c c c d d d X e e e e e e e e
f f f f f f f f

2.
f f f f f f f f b b b b b b b b
e e e e e e e e c c c c d d d a

or

a d d d c c c c b b b b b b b b
e e e e e e e e f f f f f f f f
```

Multiple choices

1. Which of the following functions will always return exactly once?

(a) exec
(b) exit
(c) fork
(d) None of the above

Solution: d)

2. Consider the following C variable declaration

```
int *(*f[3])();
```

Then f is

(a) an array of pointers to pointers to functions that return int
(b) a pointer to an array of functions that return pointers to int
(c) a function that returns a pointer to an array of pointers to int
(d) an array of pointers to functions that return pointers to int
(e) a pointer to a function that returns an array of pointers to int
(f) a pointer to an array of pointers to functions that return int

Solution: d)

Process control

What are the possible output sequences from the following program:

```
int main() {
if (fork() == 0) {
printf("a");
exit(0);
}
else {
printf("b");
waitpid(-1, NULL, 0);
}
printf("c");
exit(0);
}
```

A. Circle the possible output sequences: abc acb bac bca cab cba

Solution: abc, bac

B. What is the output of the following program?

```
pid_t pid;
int counter = 2;
void handler1(int sig) {
counter = counter - 1;
printf("%d", counter);
fflush(stdout);
exit(0);
}
int main() {
signal(SIGUSR1, handler1);
printf("%d", counter);
fflush(stdout);
if ((pid = fork()) == 0) {
while(1) {};
}
kill(pid, SIGUSR1);
waitpid(-1, NULL, 0);
counter = counter + 1;
printf("%d", counter);
exit(0);
}
```

OUTPUT: _____

Solution: 213

stack

Given the following disassembly of a short function:

```
0x00001fb6 <func+0>: push %ebp
0x00001fb7 <func+1>: movl %esp,%ebp
0x00001fb9 <func+3>: subl $0x8,%esp
0x00001fbc <func+6>: movl 0x8(%ebp),%eax
0x00001fbf <func+9>: addl %eax,%eax
0x00001fc1 <func+11>: leave
0x00001fc2 <func+12>: ret
```

and the following stack, %ebp, and %esp "right after" executing the instruction, <func+3>, at address 0x00001fb9:

```
%esp = 0x300
%ebp = 0x308
0x300: 0x00000000
0x304: 0x00000000
0x308: 0x00000328
0x30c: 0x00001fdc
0x310: 0x00000003
```

A) What is the value in %eax (the return value of the function) after we return and leave the function (after instruction <func+12>)?

Solution: The value of %eax is 6. The last instruction that touches %eax in this function is <func+9>, the add instruction. It basically doubles what was in %eax before. What was in %eax before was loaded from memory (the stack), 8 bytes after %ebp, which is 0x308 + 8 = 0x310. The value 3 is at that location, so 3 is doubled to yield 6.

B) What was the old value of the stack pointer (%esp) before we executed the first instruction of this function (at <func+0>)?

Solution: The value of %esp was 0x30c. At instruction <func+3>, we subtracted 8 bytes from the stack pointer (%esp) for temporary scratch space. That accounts for the two 0x00000000 values at the top of the stack. The instruction at <func+0> pushed the old value of %ebp onto the stack. Therefore we subtracted another 4 bytes from the stack pointer. This means that the old %esp was 12 bytes more than the current location. 0x300 + 8 + 4 = 0x30c

C) What is the old value of the base pointer (%ebp) before we entered this function?
Solution: The value of %ebp was 0x328. At instruction <func+0> (push %ebp), we pushed the old %ebp to the top of the stack by first decrementing the stack pointer to point to 0x308 (recall, it started at 0x30c). The value stored at 0x308 is the old value of %ebp.