Princeton University
COS 217: Introduction to Programming Systems
Fall 2015 Final Exam Answers

The exam was a three-hour, closed-book, closed-notes exam. Electronic devices were not allowed. Students were allowed to compose and use a one-page two-sided summary sheet.

Question 1: Grab Bag

1a) 0

*(p+2) is the first ‘b’ in “bob”, and *p is the second ‘b’ in “bob”, and ‘b’ – ‘b’ is 0.

1b) -4

```
 00...01111 ^ 00 ...01100 (XOR)
-----------------------
 00...0011 (NOT)
-----------------------
 00...11100 (one’s complement)
-----------------------
 11...11100 (two’s complement)
```

1c) Statement testing applies a set of inputs that ensure each statement executes at least once, whereas path testing must exercise all possible sequences of statements (i.e., logical paths through the code).

1d) 4

Little endian puts the “little end” first. So, the bytes are stores as 04, 03, 02, 01, with the first byte being 04. This prints as the number 4.

1e) Aliasing, when p1 and p2 point to the same location in memory.

Question 2: Building Blocks

a. C
b. P, C
c. C
d. L
e. C
f. A
g. L
h. P, L
i. C
j. A
Question 3: Totally Alarming

3a) After 11 seconds, the alarm has gone off five times (at 2, 4, 6, 8, and 10 seconds)

hi
hi
hi
hi
hi

3b) No, because alarms work on wall-clock time, not the CPU time consumed by the process. (That said, other processes might introduce minor jitter, due to the variation in the time between an alarm going off and a context switch for the process to continue running.)

3c) The user might hit control-C at the two-second mark, in between when handleAlarm() prints “hi” and the variable “done” is set to 1. If the handleControlC() handler begins running at that time, the handler would print “Okay, I won’t say hello then
, even though handleAlarm() has already printed “hi” once.

3d) The handleAlarm() signal handler needs to block the control-C signal so the control-C handler will not be invoked in the middle of handleAlarm().

3e) Sending a SIGKILL (kill -9), which cannot have its handler over-ridden.

Question 4: Be Exceptional!

a. Trap
d. Fault
c. Interrupt
g. Fault
e. Abort
f. Interrupt
h. Interrupt
i. Fault
j. Trap

Question 5: Stick a Fork in It

5a) The parent, since fork() returns the (non-zero) process-id of the child to the parent, but returns 0 to the child.

5b) The output of the first call to printf() is buffered, and does not print to the terminal before the process forks. As such, both the parent and the child have the string buffered for printing. Each process prints the string to the terminal once.

5c) Add a fflush(stdout) after the printf() and before the fork().

5d) Prints the current date/time every three seconds.
5e] If the command-line argument (argv[1]) is not a valid executable program (e.g., "/a.out foo" produces the error "/a.out: No such file or directory")

5f] The wait() system call suspends execution of the calling process until one of its children terminates.

**Question 6: Thanks for the Memories**

6a] $2^{18}$ pages

A 16 KB page size means there are $2^{14}$ bytes in a page, leaving 18 bits (32-14) for the page number.

6b] Page number: 0xEB, and Page offset: 0x3176.

The page number is the first 18 bits of 003AF176, so the 003A and the first half of F. That is, 0000 0000 0011 1010 11, or 00 0000 0000 1110 1011, or 0xEB. The page offset is the last 14 bits, so the last two bits of F (11, or 3), and the 176, or 3176.

6c] The hardware has a page table for mapping virtual addresses to physical addresses. If the mapping says that the page is stored at a particular location on disk, rather than in a physical page, the hardware generates a page fault.

6d] The program is more portable, because it can use 32-bit addresses without regard to the actual size of the physical memory, which may be (much) smaller than the virtual address space.

6e] Spatial locality: b[]

   Temporal locality: i, product

6f] Spatial locality: the i<n and product *= (they run together)

   Temporal locality: either of these two instructions (they each run frequently)

6g] Advantages: could decrease the amount of fragmentation and splitting

   Disadvantages: requires either more time (to search all memory chunks) or overhead (some sort of structure to keep memory chunks organized by size)

6h] Internal Fragmentation is when memory is wasted within allocated memory chunks, while external fragmentation is when memory is wasted between memory chunks.

**Question 7: Some Assembly Required**

7a]  
10-13
14-15
16-18
19-29
30-32
33-35
7b) func(n-1), func(n-2)

7c) It computes the nth Fibonacci number. Recursively computes Func(n) = func(n-1) + func(n-2) where f(0) = 0 and f(1) = 1.

7d)

```
pushq  VAR1   OR  pushq, %rbx
pushq  %rdi
popq   %rdi
popq   VAR1   OR   popq, %rbx
```

7e) callee-saved: %rbx    caller-saved: %rdi, %rax

7f) scanf takes in a memory address as a parameter, so we need to access the address of the local variable i and it therefore must exist in memory.

7g) pushes %rip on to the stack
    jumps to the func memory address

**Question 8: Going Modular: Mind your C and Queues**

8a) Abstract Object

8b)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing protection against multiple</td>
<td>Add #ifndef, #define, #endif</td>
</tr>
<tr>
<td>inclusions</td>
<td></td>
</tr>
<tr>
<td>State of the queue declarations are in the</td>
<td>Place the declarations in the module’s</td>
</tr>
<tr>
<td>interface and non-static, which allows the</td>
<td>implementation and make them static</td>
</tr>
<tr>
<td>client to access the module’s fields</td>
<td></td>
</tr>
</tbody>
</table>

8c) We need to have enough space for all of the author characters as well as for the null character to mark the end of the author string.

8d) line_token: 'T' at the beginning of "To teach ..."
    start_token: 'J' at the beginning of "Joubert: ..."
    end_token: ':' at the beginning of ": To teach..."

8e) -   strtok delimits on characters, and we are looking for the string ": " (colon space)
       OR
-   With strtok you cannot parse different strings concurrently as strtok uses static storage to keep track of the current string position between calls.
8f)
DATA
Stack
RODATA
Stack
HEAP

8g)
Quote.h:
#ifndef QUOTE_INCLUDED
#define QUOTE_INCLUDED

typedef struct Quote *Quote_T;

Quote_T Quote_new(char* line_token);
void Quote_free(Quote_T quote);
void Quote_print(Quote_T quote);

#endif

Quote.c:
#include quote.h

lines 6-9 (quote structure type definition)

struct Quote* Quote_new(char* line_token) {
    Lines 21-34 (all the lines in while loop except last two)
    return quote;
}
void Quote_free(Quote_T q){
    Lines 42-44 (3 calls to free)
}
void Quote_print(Quote_T q){
    Line 41
}

Question 9: More Assembly Required: Fun with Structs!

9a) val: 0  po: 8

9b) A singly linked (non-circular) list of structures of type struct point

9c) Multiplies all values in a linked list of point structs.

long fun(struct point *p){
    long n = 1;
    while (p != NULL) {
        n *= ptr->val;
        p = p->po;
    }
    return n;
}