0/35 Questions Answered

TIME REMAINING
1 hr 49 mins

COS 217 Final Exam

Q1 Be our guest - put our exam to the test
0 Points

This exam consists of 7 substantive questions (Q2-Q8 -- Q1 is only exam information and Q10 is only the Honor pledge) totaling 90 points. Q9 is a 1-point lecture "attention to detail" extra credit. Most of the substantive questions are made up of multiple parts, with points allocated as indicated.

Unless you have confirmed ODS accommodations, you have 3 hours (180 minutes) to complete the exam from the time you begin. Unless you have received an alternate exam window from Dr. Moretti via your Dean or Director of Studies, you must complete the exam by the end of the day (11:59 PM Princeton time (US Eastern Time)) on December 12, irrespective of when you began.

In Gradescope, students' answers are autosaved as they enter them. We have observed a couple second latency, though, so we advise against changing answers right up to the deadline. There is a countdown timer (which can be hidden) in the top right corner of the screen.

This exam is "open-book" but "closed-communication":

- You are not allowed to communicate with any other person, whether inside or outside the class. You may not send the exam problems to anyone, nor receive them from anyone, nor communicate any information about the problems or their topics.
- You are allowed to consult any material from course lectures, precepts, readings, assignments, Ed, etc.

- You are allowed to use resources found on the web, so long as they do not violate the communication rule above (i.e., so long as they are not solicited by you). As an example, you can read an old Stack Overflow post, but you can't post a question to Stack Overflow.
- You may build and run any code on armlab (though be careful, as this can be a dramatic time sink!)

You may post (private!) posts to Ed to seek clarification from the course staff. We will monitor Ed regularly, however we cannot guarantee 24-hour availability throughout the exam period.

This examination is administered under the Princeton University Honor Code. All suspected violations of the Honor Code must be reported to the Committee on Discipline. You will attest to the standard pledge in Q10 after you finish your responses.

○ I've read this. Okay!

Q2 Be Bashful

9 Points

Q2.1

5 Points

Select all of the following options that will produce an empty file (that is, a file with length 0) called `myfile` when executed in `bash` (assume that `myfile` does not already exist):

☐ `echo > myfile`

☐ `touch myfile`

☐ `:> myfile`

☐ `true || cp /dev/null myfile`

☐ `>| myfile`

Q2.2
2 Points

You have two programs, `first` and `second`, both of which appear in your `PATH`. You would like to execute `first` and then execute `second` only if `first` succeeds. Write 1-2 lines of `bash` commands that will accomplish this:

Enter your answer here

Q2.3
2 Points

You have several `bash` commands that you'll be using over and over. Give two plausible ways to improve this workflow versus typing in the commands repeatedly:

Enter your answer here

Q3 You're no Dumbo when you automate building and testing.

13 Points

Consider the following C code, which is executed only once in its program. You may assume that `x`, `y`, and `z` are initialized with `int` values from standard input before executing this code:

```
switch (x) {
   case 1: y *= foo(y);
           y++;
   case 2: y *= bar(y);
           y--;
   default:
           y *= baz(y);
}

if (z)
   y++;
else
   y--;
```

Q3.1
2 Points

How many input data sets will be necessary in order to achieve complete statement testing of this code?

Enter your answer here

Q3.2
2 Points

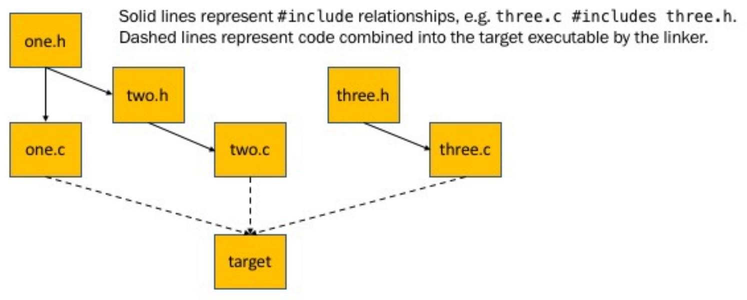
How many input data sets will be necessary in order to achieve complete path testing of this code?

Enter your answer here

Q3.3
9 Points

Create a proper `Makefile` for the dependency graph below. Your

`Makefile` should minimize the number of recompilations necessary by supporting partial-builds based on the dependency graph. You are not required to have any non-file targets, however the final executable `target` must be created when `make` is invoked with no arguments.


Solid lines represent #include relationships, e.g. three.c #includes three.h.
Dashed lines represent code combined into the target executable by the linker.
one.h
two.h
three.h
one.c
two.c
three.c
target

Note -- Gradescope will not let you enter a tab character in the response field, so instead you can use 8 spaces as the prefix to the command for any rule.

Enter your answer here

Q4 Li Shang'll make a string out of you

12 Points

For each part of this question you are presented with three code snippets. Assume each snippet is in its own one-file program with all appropriate header files included. In each snippet, lines prefixed with `o:` are outside any function and lines prefixed with `f:` are within a function and appear contiguously in the order given.

You will indicate which of the snippets result in the specified memory section containing the specified data. It could be none of them, all of them, or anywhere in between.

For example, here is an example of a snippet that results in the RODATA section containing the 7 bytes of the string "RODATA":

```
f: puts("RODATA");
```

It is okay if the snippet also results in another section containing the specified data.

Q4.1

3 Points

Consider the following code snippets:

```
/* A */
f: char stack[] = {'S', 'T', 'A', 'C', 'K', '\0'};
```

```
/* B */
f: char *stack = "STACK";
```

```
/* C */
f: char stack[6];
f: strcpy(stack, "STACK");
```

Which of these snippets result in the STACK section containing the 6 bytes of the string "STACK"?

- ☐ A
- ☐ B
- ☐ C

Q4.2

3 Points

Consider the following code snippets:

```
/* D */
f: char *heap = malloc(strlen("HEAP")+1);
f: heap = "heap";
```

```
/* E */
f: char *heap = calloc(5, sizeof(char));
f: char *pile = "HEAP";
f: for (i = 0; i < 4; i++) /* assume i has been declared as a size
f:     heap[i] = pile[i];
```

```
/* F */
f: char **heap = malloc(sizeof(char*));
f: heap[0] = strcpy(malloc(5), "HEAP");
```

Assuming memory allocation always succeeds, which of these snippets result in the HEAP section containing the 5 bytes of the string "HEAP"?

- ☐ D
- ☐ E
- ☐ F

Q4.3
3 Points

Consider the following code snippets:

```
/* G */
o: char data[5] = "DATA";
```

```
/* H */
o: char *data;
```

```
f: data = "DATA";
```

```
/* I */
o: char *data = "217!";
f: data = "DATA";
```

Which of these snippets result in the DATA section containing the 5 bytes of the string "DATA"?

- ☐ G
- ☐ H
- ☐ I

Q4.4
3 Points

Consider the following code snippets:

```
/* J */
o: char *bss;
f: bss = "BSS";
```

```
/* K */
o: char bss[4];
f: strcpy(bss, "BSS");
```

```
/* L */
o: char *bss = NULL;
f: strcpy(bss, "BSS");
```

Which of these snippets result in the BSS section containing the 4 bytes of the string "BSS"?

☐ J

☐ K

☐ L

Q5 Soul -> Onward -> Frozen II -> ... -> Snow White

21 Points

Consider the following buggy implementation of a `List` construct, and assume that all necessary interface files have been included:

```
/* a Node_T is a member of the List with a string as contents */
typedef struct node* Node_T;

/* building block of the List */
struct node {
   /* contents of node */
   char* payload;
   /* next node in List */
   Node_T next;
};

/* head of the List */
static Node_T first = NULL;

/* if payload is not already in the List,
   inserts a new node at front of the List having contents payloac
   returns 1 if insertion is successful, 0 if unsuccessful. */
int List_insert(const char* payload) {
   Node_T curr = first;
   assert(payload != NULL);
   while(curr != NULL)
      if(!strcmp(curr->payload, payload))
         return 0;
   curr = malloc(sizeof(struct node));
   if(curr == NULL)
      return 0;
   curr->next = first;
   curr->payload = malloc(strlen(payload)+1);
   if(curr->payload == NULL)
```

```
        return 0;
     strcpy(curr->payload, payload);
     return 1;
  }

  /* removes all nodes from the List */
  void List_free() {
     Node_T current;
     for(current = first; current != NULL; current = current->next)
        free(current);
  }
```

Q5.1
1 Point

`List`, as defined and used here, is a(n):

- ○ Stateless Module
- ○ ADT
- ○ AO
- ○ None of these

Q5.2
1 Point

`Node_T`, as defined and used here, is a(n):

- ○ Stateless Module
- ○ ADT
- ○ AO
- ○ None of these

Q5.3
1 Point

Making a defensive copy of the payload string in `List_insert` is unnecessarily cautious, because the `payload` parameter to `List_insert` is declared `const`.

○ True

○ False

Q5.4
9 Points

Identify three bugs in the `List_insert` function and how each could be fixed.

A bug for this problem is something that causes a warning or error from gcc217, a runtime crash, behavior that violates the function's contract, or a dynamic memory management issue observable by MemInfo or Valgrind.

Bug 1:

Enter your answer here

Bug 2:

Enter your answer here

Bug 3:

Enter your answer here

Q5.5
9 Points

Identify three bugs in the `List_free` function and how each could be fixed.

A bug for this problem is something that causes a warning or error from gcc217, a runtime crash, behavior that violates the function's contract, or a dynamic memory management issue observable by MemInfo or Valgrind.

Bug 1:

Enter your answer here

Bug 2:

Enter your answer here

Bug 3:

Enter your answer here

Q6 Ursula implores you to "Go ahead! Make your choice!"

10 Points

Here are the C definitions for a slightly different list from the one in the previous Question:

```
/* a Node_T is a member of a collection holding unsigned long valu
typedef struct node* Node_T;

struct node {
   /* contents of node */
   unsigned long payload;
   /* next node in the list */
   Node_T next;
};

struct list {
   /* head of the list */
   Node_T first;
   /* number of nodes in the list */
   unsigned long length;
};

/* a List_T is a collection of unsigned longs */
typedef struct list* List_T;
```

And here is an AARCH64 assembly language function correctly implementing some operation for such a list:

```
        .global List_mystery
List_mystery:
        sub     sp, sp, 32
        str     x0, [sp,8]
        ldr     x1, [x0,8]
        cmp     x1, xzr
        bne     .L2
        mov     x0, 0
        b       .L3
.L2:
        ldr     x1, [x0]
        ldr     x2, [x1]
        str     x2, [sp,16]
        str     x1, [sp,24]
        ldr     x2, [x1,8]
        b       .L4
.L6:
        ldr     x3, [x2]
        ldr     x0, [sp,16]
        cmp     x3, x0
        bls     .L5
        str     x3, [sp,16]
        str     x2, [sp,24]
.L5:
        ldr     x2, [x2,8]
.L4:
        cmp     x2, xzr
        bne     .L6
        ldr     x0, [sp,24]
.L3:
        add     sp, sp, 32
        ret
```

Answer each part of this question based on the code above.

Q6.1
1 Point

How many parameters does the function `List_mystery` take?

- ○ 0
- ○ 1
- ○ 2
- ○ 3 or more

Q6.2
1 Point

Does the function `List_mystery` appear to return a value?

- ○ Yes
- ○ No

Q6.3
1 Point

What is the purpose of lines 13-14:

```
str x2, [sp, 16]
str x1, [sp, 24]
```

- ○ Save the value of some parameters
- ○ Save the value of some callee-saved registers
- ○ Allocate space for some local variables
- ○ Initialize some local variables

Q6.4
2 Points

Label `.L6` on line 17 begins the body of a loop. Which is the last instruction corresponding to the loop's body in the C code? Note: a C loop's body does not include the `for(...)` or `while(...)` portion.

- ○ Line 23: `str x2, [sp, 24]`
- ○ Line 25: `ldr x2, [x2, 8]`
- ○ Either Line 23 or Line 25, depending on whether the C loop is a `for` or `while` loop.
- ○ Line 28: `bne .L6`
- ○ None of the above.

Q6.5
1 Point

Lines 20 and 21 are comparing values corresponding to which C variable type?

- ○ `struct node`
- ○ `unsigned long`
- ○ `Node_T`
- ○ `struct list`
- ○ `List_T`

Q6.6
1 Point

What is the purpose of Line 29 (`ldr x0, [sp,24]`)?

- ○ Copy the return value
- ○ Clean up a local variable
- ○ Restore a caller-saved register
- ○ Restore a callee-saved register

Q6.7
1 Point

Nothing is stored at [SP, 0], thus a reasonable space optimization would be to move the data stored at stack offsets 8, 16, and 24 to 0, 8, and 16, instead, allowing us to change the first and penultimate instructions of the function to `sub sp, sp, 24` and `add sp, sp, 24`, respectively.

- ○ True
- ○ False

Q6.8
2 Points

The function `List_mystery` doesn't save or restore the value of `x30`. Choose the **best** answer for why this is okay:

- ○ The function doesn't use the stack
- ○ The function doesn't use callee-saved registers
- ○ The function doesn't return a value
- ○ The function isn't recursive
- ○ The function doesn't call any other functions

Q7 Like Yen Sid from Fantasia.
16 Points

This question will again be dealing with the code for `List_mystery` from the previous Question, repeated here for convenience:

```c
/* a Node_T is a member of a collection with an unsigned long as c
typedef struct node* Node_T;

struct node {
   /* contents of node */
   unsigned long payload;
   /* next node in the list */
   Node_T next;
};

struct list {
   /* head of the list */
   Node_T first;
   /* number of nodes in the list */
   unsigned long length;
};

/* a List_T is a collection of unsigned longs */
typedef struct list* List_T;
```

```
        .global List_mystery
List_mystery:
        sub     sp, sp, 32
        str     x0, [sp,8]
        ldr     x1, [x0,8]
        cmp     x1, xzr
        bne     .L2
        mov     x0, 0
        b       .L3
.L2:
        ldr     x1, [x0]
        ldr     x2, [x1]
        str     x2, [sp,16]
        str     x1, [sp,24]
        ldr     x2, [x1,8]
        b       .L4
.L6:
        ldr     x3, [x2]
        ldr     x0, [sp,16]
        cmp     x3, x0
        bls     .L5
        str     x3, [sp,16]
        str     x2, [sp,24]
.L5:
        ldr     x2, [x2,8]
```

```
.L4:
        cmp     x2, xzr
        bne     .L6
        ldr     x0, [sp,24]
.L3:
        add     sp, sp, 32
        ret
```

Q7.1
16 Points

Translate the `List_mystery` function into "Flattened C", using the same labels as the given assembly language code. Include a function comment for `List_mystery` that meets the requirements from your programming assignments in this course.

Enter your answer here

Q8 Avengers Disassemble!
9 Points

Consider the following `objdump` output:

```
armlab02$ objdump --disassemble --reloc simple.o

simple.o:     file format elf64-littleaarch64


Disassembly of section .text:

0000000000000000 <main>:
   0:   a9be7bfd        stp     x29, x30, [sp,#-32]!
   4:   910003fd        mov     x29, sp
   8:   90000000        adrp    x0, 0 <main>
                        8: R_AARCH64_ADR_PREL_PG_HI21    .rodata
   c:   91000000        add     x0, x0, #0x0
                        c: R_AARCH64_ADD_ABS_LO12_NC     .rodata
  10:   f9000fa0        str     x0, [x29,#24]
  14:   f9400fa0        ldr     x0, [x29,#24]
  18:   94000000        bl      0 <strlen>
                        18: R_AARCH64_CALL26    strlen
```

```
    1c:   aa0003e1        mov     x1, x0
    20:   90000000        adrp    x0, 0 <main>
                          20: R_AARCH64_ADR_PREL_PG_HI21  .rodata+0x
    24:   91000000        add     x0, x0, #0x0
                          24: R_AARCH64_ADD_ABS_LO12_NC   .rodata+0x
    28:   94000000        bl      0 <printf>
                          28: R_AARCH64_CALL26    printf
    2c:   52800000        mov     w0, #0x0                          //
    30:   a8c27bfd        ldp     x29, x30, [sp],#32
    34:   d65f03c0        ret
```

Q8.1
1 Point

Does this output represent the full contents of `simple.o`?

- Yes
- No

Q8.2
1 Point

What is `0:` on the line beginning with that same prefix?

- Assembly language code
- Machine language code
- Address
- Offset
- Relocation record
- Data value

Q8.3
1 Point

What is `910003fd` on the line prefixed with `4:`?

- ○ Assembly language code
- ○ Machine language code
- ○ Address
- ○ Offset
- ○ Relocation record
- ○ Data value

Q8.4

1 Point

What is `R_AARCH64_ADR_PREL_PG_HI21 .rodata` on the (indented) second line prefixed with `8:`?

- ○ Assembly language code
- ○ Machine language code
- ○ Address
- ○ Offset
- ○ Relocation record
- ○ Data value

Q8.5

1 Point

Which software produced the `R_AARCH64_ADR_PREL_PG_HI21 .rodata` on the (indented) second line prefixed with `8:`?

- ○ Preprocessor
- ○ Compiler
- ○ Assembler
- ○ Linker
- ○ Standard Library

Q8.6

1 Point

Which software is the intended consumer of the `R_AARCH64_ADR_PREL_PG_HI21 .rodata` on the (indented) second line prefixed with `8:`?

- ○ Preprocessor
- ○ Compiler
- ○ Assembler
- ○ Linker
- ○ Standard Library

Q8.7

2 Points

Resolving a `R_AARCH64_CALL26` construct could change which byte(s) in the corresponding `bl` instruction? For this question, bytes are numbered 0 (least significant) through 3 (most significant).

- ○ All four (bytes 0-3)
- ○ The three least significant (bytes 0-2)
- ○ The two least significant (bytes 0-1)
- ○ The three most significant (bytes 1-3)
- ○ The two most significant (bytes 2-3)
- ○ Only a single byte
- ○ No change will be made by processing `R_AARCH64_CALL26`

Q8.8
1 Point

While resolving a `R_AARCH64_CALL26` construct, from where will the machine code defining `strlen` or `printf` be sourced?

- ○ `simple.h`
- ○ `simple.c`
- ○ `simple.s`
- ○ `simple.o`
- ○ `libc.a` or `libc.so`
- ○ `string.h` or `stdio.h`, respectively
- ○ `strlen.o` or `printf.o`, respectively

Q9 Dr. Moretti, not Rapunzel, has hidden it -- somewhere you'll never find it.
1 Point

Note any easter egg (sub-second flash image with a pop culture or historical reference) from any COS 217 lecture video. As an alternative, list any similarly tortured stretch of a pop culture or historical reference given in a lecture's narration, even if it did not make an appearance on the slides. (Feel free to list as many as you noticed and recall -- for better or worse! -- but a single one will do for the bonus point.)

Enter your answer here

Q10 The wonderful thing about Honor Codes is Honor Codes are wonderful things

0 Points

Copy the Honor pledge in the field below:

I pledge my honor that I have not violated the Honor Code during this examination.

Enter your answer here

Enter your name in the field below, attesting to the Honor pledge you have copied above:

Enter your answer here

Submit & View Submission