# Midterm Exam  Spring 2025



This exam consists of 4 substantive questions. You have 50 minutes – budget your time wisely. Assume the ArmLab/gcc217 environment unless otherwise stated in a problem.

Do all of your work on these pages. You may use the provided blank spaces for scratch space, however this exam is preprocessed by computer, so for your final answers to be scored you must write them inside the designated spaces and fill in selected circles and boxes completely (⚫ and ⬛, not ✔ or ✘). Please make text answers dark and neat.

Name: **Sample Solutions** NetID:

Precept:

- ◯ P01 - MW 1:30 Xiaoyan Li
- ◯ P02 - MW 3:30 Xiaoyan Li
- ◯ P03 - TTh 12:30 Lana Glisic
- ◯ P04 - TTh 12:30 Tolulope Oshinowo
- ◯ P06 - TTh 1:30 Indu Panigrahi
- ◯ P07 - TTh 1:30 Ryan Oet
- ◯ P08 TTh 3:30 Yang Duan
- ◯ P09 TTh 7:30 Andrew Sheinberg

This is a closed-book, closed-note exam, except you are allowed one one-sided study sheet. Please place items that you will not need out of view in your bag or under your working space at this time. Electronic devices such as cell phones, laptops, smartwatches except to check the time, etc. may not be used during this exam.

This examination is administered under the Princeton University Honor Code. Students should sit one seat apart from each other and refrain from talking to other students during the exam. All suspected violations of the Honor Code must be reported to honor@princeton.edu.

In the box below, copy **and** sign the Honor Code pledge before turning in your exam:
*"I pledge my honor that I have not violated the Honor Code during this examination."*

**Exam Statistics**
**(out of 50 possible points)**

**Maximum: 47**
**Mean: 30.76**
**Median: 30**
**Standard Deviation: 7.3**

**Percentiles:**
**10th = 21**
**20th = 24**
**30th = 27**
**40th = 29**
**50th = 30**
**60th = 33**
**70th = 35**
**80th = 37**
**90th = 40**

X_____

## Question 0: Dear Friends, 0 points

Please don't make the course staff's life harder: make sure you have filled out your name, NetID (i.e., armlab login – not PUID, not email alias), precept and the Honor Code pledge text on the front page. Sign your name once you have finished the exam.

## Question 1: Portability after Retirement from the EQuad 5 points

For each of the code snippets below, indicate whether the equality operation always evaluates to true (`1`), always evaluates to false (`0`), or depends on the system because the behavior is not guaranteed by the C standard.

| | | TRUE | FALSE | DEPENDS |
|---|---|---|---|---|
| a. | `031 == 31`<br>Integer literals beginning with a leading 0 are in octal. Octal 31 = decimal 25, not 31. Both of these numbers are guaranteed to fit inside an int, so there's no portability concern. | ○ | ● | ○ |
| b. | `sizeof(struct {long seas; int leader;}) == sizeof(long) + sizeof(int)`<br>The amount of padding within a struct is compiler-dependent based on architecural alignment requirements/preferences. | ○ | ○ | ● |
| c. | `enum { BSE }; BSE == 0`<br>The C90 standard states that if the first enumerator has no = assignment, the value of its enumeration constant is 0. | ● | ○ | ○ |
| d. | `int princeton, inn; princeton = inn = 1983; princeton++ == ++inn;`<br>princeton++ and ++inn both increment their variables to 1984, but the value emitted differs: post-increment evaluates to the old value (1983), while pre-increment evaluates to the new value (1984).<br>Portability of the value is no a concern, because an int must be at least 2 bytes, and thus must have a max value of at least 32768. | ○ | ● | ○ |
| e. | `int peter; int bogucki; &bogucki == &peter + 1`<br>Where local variables are placed on the stack relative to each other is compiler-dependent. | ○ | ○ | ● |

(Hint: recall pointer addition is element-wise: `+ 1` is really `+ sizeof(int)` bytes)

## Question 2: Unique administrator 13 points

The C program on the next page is intended to implement a subset of the Linux `uniq` tool, equivalent to `uniq -c`, by printing out each line from `stdin` but suppressing subsequent adjacent duplicate lines and prepending each printed line with the total number of adjacent copies. For example:

| Input | Output |
|---|---|
| Going back,<br>Going back,<br>Going back to Nassau Hall.<br><br>Going back,<br>Going back,<br>To the best old place of all. | 2 Going back,<br>1 Going back to Nassau Hall.<br>1<br>2 Going back,<br>1 To the best old place of all. |

```c
#include <string.h>
#include <stdio.h>
int main(void) {
   enum {MAX_LINE_SIZE = 1024};
   char acLine1[MAX_LINE_SIZE], acLine2[MAX_LINE_SIZE];
   char *pcLineNew = acLine1;
   char *pcLineOld = acLine2;
   size_t uRepeatCount = 0;
   /* for each line of standard input */
   while(fgets(pcLineNew, MAX_LINE_SIZE, stdin) != NULL)
      /* if the new line matches the prior line */
      if(①) uRepeatCount++;
      else { /* print old line, if needed, and reset for new line */
         if(②) printf("%lu %s", uRepeatCount, pcLineOld);
         uRepeatCount = ③;
         /* Hint: ④ and ⑤ update the value of pcLineNew. */
         pcLineOld = pcLineNew;
         if(pcLineNew == acLine1) ④;
         else ⑤;
      }
   if(②) printf("%lu %s", uRepeatCount, pcLineOld);
   return 0;
}
```

There are five expressions (one of which appears twice) that have been omitted. In the boxes below, fill in expressions that will correctly implement the specification on page 2:

a. ① **strcmp(pcLineNew, pcLineOld) == 0**

This compares the entire contents of the two strings, which is what we need — not their pointers, not their first character, not their string length, etc.

b. ② **uRepeatCount != 0**

This is a check to avoid printing the uninitialized acLine2 during the first loop iteration.
Alt answers: uRepeatCount > 0, RepeatCount >= 1, uRepeatCount (since 0 is false/non-0 true)

c. ③ **1**

This line resets uRepeatCount to start counting repetitions of the new string that didn't match. We have seen one instance already when we reset, so the count is reset to 1.

d. ④ **pcLineNew = acLine2**

This conditional swaps which of the two arrays (acLine1 or acLine2) pcLineNew is pointing to.

(pcLineOld is correspondingly swapped by the assignment on the previous line.)

⑤ **pcLineNew = acLine1**

This allows us to always read into pcLineNew without ever having to copy the entire string contents of pcLineNew into pcLineOld.

## Question 3: Best Advising: Review and Correct — 16 points

Each of the four functions below has one or more bugs. Each bug may be corrected by replacing one line from the existing code or adding some extra code to the function. For replacements, cross out the incorrect line in the box on the left and write the corrected line in the box on the right. For insertions, add an arrow in the box on the left and write the code to be inserted in the box on the right. Do not re-write lines that are already correct. Assume that all necessary headers have been included.

For example:

```
/* Print a plus b, return 0. */
int printSum(int a, int b) {
                              <--------
   return a + b;
}
```

```
printf("%d", a + b);
return 0;
```

a.

```
/* Return whether y is strictly
   between x and z. */
int twixt(int x, int y, int z) {
   return x <= y <= z;
}
```

There were two changes that needed to be made:

1. change from <= to < to correctly consider "strictly" between
2. combine two separate conditions instead of 1 logically incorrect chained condition.

```
return (x < y) && (y < z);
```

b.

```
/* return the location of the
first character in s1 that
differs from its counterpart in
s2, or NULL if the strings'
contents are the same */
char *diverge(const char* s1,
              const char* s2) {
   assert(s1 != NULL);
   assert(s2 != NULL);
   while(*s1 != '\0')
      if(*s1 == *s2) {
         s1++; s2++;
      }
      else
        return s1;
   return NULL;
}
```

There were two changes that needed to be made:

1. fix the type mismatch on the first return by casting away constness
2. handle the case where s1 is a proper prefix of s2:
   the function should return the address of the '\0' at the end of s1

```
return (char *) s1;
if(*s2 != '\0') return (char *) s1;
```

c.

```
typedef struct Node* Node_T;
struct Node { Node_T next; };

/* return whether n is the
second-to-last node (i.e., the
node immediately before the last
node) in its list */
int penultimate(Node_T n) {
   assert(n != NULL);
   assert(n->next != NULL);
   return n->next->next == NULL;
}
```

If n is the last node in the list, the second assert will fail, which is not the correct function behavior of returning 0.

if( n->next == NULL) return 0;

d.

```
/* expand ai, which has been
dynamically allocated with n
elements, to be twice as long.
Return the expanded array, or
NULL if memory allocation
returns NULL. */
int *twice(int ai[], size_t n) {
   assert(ai != NULL);
   ai = realloc(ai, 2 * n);
   return ai;
}
```

realloc's second parameter should be a number of bytes, not a number of elements, but the parameter n is a number of elements, so this is off by a factor of the size of an int.

ai = realloc(ai, 2 * n * sizeof(int));

(The exam continues with Question 4 on page 6.)

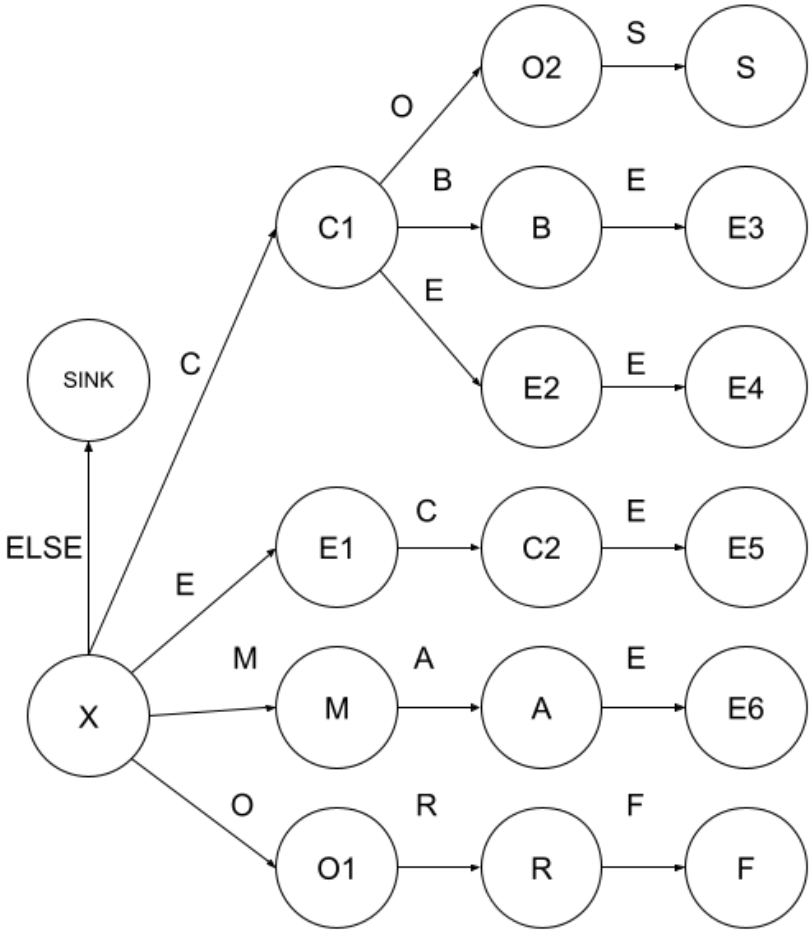
## Question 4: Departmentistic Finite Automaton (FSA) 16 points

The admissions office wants to forward questions from BSE applicants to their potential department. To do so, they've asked a COS 217 student to build a program that finds messages that include the 3-letter strings that correspond to the six SEAS department abbreviations (CBE, CEE, COS, ECE, MAE, and ORF).

The student has modeled this as a DFA. For each message, they will run each word (one at a time, resetting between each) through the DFA until either a word is accepted, at which time they will send the question to the named department, or until all words in the message have been rejected.

Consider the DFA below, which should accept **only** 3-letter strings that correspond to one of the six relevant department abbreviations. Assume that all transitions, if not explicitly shown on the DFA, transition into the **SINK** state.

Questions about this DFA appear on page 7.



For each question below, answer in the box provided.

a. Which is the start state? Identify it by name, e.g. SINK or E6.

X is the only state that has no incoming transitions, and thus must be the start state.

X

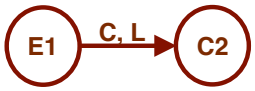
b. What are all the accepting states? Identify them by name.

The six states on the right column in the DFA correspond to the last letter in the path of each department.
Thus, these states are the accepting states, since if a string stops there, it indicates that it was exactly a 3-letter string matching a department.

S, E3, E4, E5, E6, F

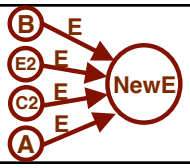
c. ECE changed name from ELE in 2021. What is the smallest change to the DFA that would allow forwarding messages to ECE that have either ECE or ELE? Answer in a sentence or draw any new/updated nodes and transitions in the box.

Change the E1-to-C2 transition to transition on either C or L.




d. This DFA has 18 states (including the sink), but it can be done in 15. In one sentence, clearly describe what changes would have to be made to do so. If you prefer, instead of a sentence, you may draw the correct 15-state version on page 8 and state in the box that you have done so.

The simplest correct approach is to combine the accepting states E3, E4, E5, and E6 into a single accepting state that transitions on E from B, E2, C2, and A.




e. The 15 state version is equivalent to the 18 state version in terms of the strings it accepts, but it has a key drawback if one were to use it for the question forwarding purpose outlined above. In the box, describe in at most two sentences why the smaller DFA is not suitable as the underlying design for this application:

In the original DFA, each accepting state represented a unique accepted string.

When we combine accepting states, if we end in that state it still means that we found a department, but we can't uniquely tell which department is the appropriate recipient because a DFA has no "memory" of its prior input that resulted in arriving at its current state.

(Question 4 was the last question. The space below is intentionally left blank. You may use it for scratch work, however any answers given below will not be graded, except if you stated in Question 4 that you would draw your answer on this page.)