Final Exam Grading Information

Below you can find information about our grading criteria and process for each final exam problem. Where applicable, deductions were capped at the number of points for that part.

1. Floats (25)

A and B:

We awarded 5 points for each of these two parts. We awarded partial credit in 2 scenarios: if the sign bit appeared to have been flipped in the calculation (e.g. -64 instead of 64 for A), or if the exponent was off by one (e.g. calculated as 2^7 instead of 2^6). These were the only cases we chose to award partial credit.

C: We awarded 7 points for this problem. We had a -4 deduction each for errors with the advantage 32-bit signed ints have over floats and the disadvantage 32-bit signed ints have over floats. We had 1 partial credit deduction for each part for small errors, such as including an incorrect number range when discussing the representations, or if the wording was slightly ambiguous (e.g. "ints can represent all numbers from INT_MIN to INT_MAX", when they can represent all *integers* within these values). More significantly incorrect answers did not receive credit.

D: We awarded 2 points for each correct output line and did not award partial credit.

2. Strings and Memory (35)

We awarded 4 points for blank 1, 5 points each for blanks 2-4, and 16 points for blank 5, split across the 3 lines (6 points for resizing, 5 points each for shifting memory and copying in the new string). Here is some information about how we graded each blank. All blanks except for blank 1 used a "major/minor" grading scale, meaning we used the following scale:

Correct (full credit)

Minor Error (partial credit)

Major Error / 2+ Minor Errors (no credit)

Blank 1

We did not award partial credit for this problem.

Blank 2

Here are some examples of errors that we deemed minor and major:

Minor:

- Comparison operator mistake
- No dereference
- Off-by-one in math calculation

Major:

- Incorrect pointer comparison (e.g. comparing to NULL)

Blank 3

Here are some examples of errors that we deemed minor and major:

Minor:

- Incorrect level of indirection
- Incorrect checking of return value of strcmp
- Used strncmp instead of strcmp

Major:

- No strcmp-or-similar string comparison function/operation used

Blank 4

Here are some examples of errors that we deemed minor and major:

Minor:

- Incorrect level of indirection
- Off-by-one error
- = instead of +=

Major:

- No strlen-or-similar (e.g. used sizeof instead of strlen)

Blank 5, line 1 (resizing):

Here are some examples of errors that we deemed minor and major:

Minor:

- Didn't store return value of realloc
- Issue with level of indirection of parameter/return value
- Off-by-one error
- Didn't realloc string_list_ptr (e.g. realloc'ed string_list_curr)

Major:

- No call to realloc
- Only pass a size parameter to realloc
- No strlen-or-similar operation to include length (e.g. used sizeof instead of strlen)

Blank 5, line 2 (shifting memory)

Here are some examples of errors that we deemed minor and major:

Minor:

- Issue with level of indirection of parameter
- Off-by-one error
- Used memcpy instead of memmove
- Didn't move string_list_ptr (e.g. moved string_list_curr)

Major:

- No call to memcpy or memmove
- No strlen-or-similar operation to include length (e.g. used sizeof instead of strlen)

Blank 5, line 3 (copying in new string)

Here are some examples of errors that we deemed minor and major:

Minor:

- Issue with level of indirection of parameter
- Off-by-one error
- No/incorrect handling of null terminator

Major:

- Used strcat
- Used direct assignment of pointers instead of a copying function like strcpy

3. Assembly (40)

A: Reverse Engineering

We graded each blank out of 2-3 points (for a total of 25), with limited deductions that awarded partial credit. Specifically:

Blank 1 (i + *x): 3 points, 1/3 for not dereferencing x

Blank 2 (107): 2 points

Blank 3 (1): 2 points

Blank 4 (max): 3 points

Blank 5 (i *= -2): 3 points, 1/3 for omitting negative sign or assignment (e.g. i * -2)

Blank 6 (i < 0): 3 points

Blank 7 (bar(i, &x)): 3 points, -1 for errors in each of the function, first parameter and second parameter

Blank 8 (counter): 3 points

Blank 9 (sum + 107 or sum + x): 3 points

Miscellaneous deduction: -1 for using hexadecimal instead of decimal for 107 (the problem statement required decimal notation)

For the if/else statement, we awarded full credit if the condition was flipped (i.e. i >= 0) as long as the corresponding if/else body statements were flipped as well. We gave full credit for blanks 7 and 8 as long as both of the listed answers appeared exactly once.

B: We awarded 5 points for this problem, 2 points for correctly specifying the location of x, and 3 points for specifying why x must live there. We did not award further partial credit. Note that we also accepted answers that interpreted "why must it live there?" as "justify your answer" instead of "why is it required to live there".

C: We awarded 5 points for this problem, 2 points for correctly specifying the purpose of such a test instruction, and 3 points for specifying its impact on a following jns instruction. We did not award further partial credit. We gave full credit for any answer for the first part of C that described the general correct behavior for test with two of the same argument, including answers like "it checks the sign of the value" or "it sets flags based on the value of the argument", but did not give full credit for answers that omitted significant information such as "it sets the sign flag based on the value of the argument" (since it also sets the zero flag).

D: We awarded 5 points for this problem, 2 points for correctly specifying the pushed registers are caller-owned, and 3 points for specifying why these registers must be pushed. We did not award further partial credit.

4. Gaming The System (30)

A: We awarded 8 points for this problem, 4 points each for specifying the value of i and &scores_copy[i]. We awarded partial credit for solutions to &scores_copy[i] with the correct address but an incorrect type (e.g. int * instead of unsigned int *). We also accepted answers that specified the values of these two expressions at the *end* of the first loop iteration, which would be 0 and 0xfffffffe920. We did not award any additional partial credit.

B: We graded this problem out of 4 points and did not award any partial credit.

C: We graded this problem out of 12 points, 6 points for discussing overwriting the loop variable i and 6 points for discussing how the loop does not iterate over/change the scores. Within each of these categories, we awarded full (6), partial (3) or no (0) credit. Partial credit was awarded for minor errors or mistakes, while more significant incorrect statements did not receive any credit.

D: We graded this problem out of 6 points, 2 points each for mentioning the line of code to change, the specific modification that should be made, and the justification for why this change resolves the issue. We did not award further partial credit.

5. Heap Allocators (50)

Each part of this problem was out of 5 points, with the exception of malloc_within_zone, which was out of 15 points.

A: We awarded 1 point for obtaining the payload_size_and_status field, 2 points for & with 2 (or equivalent), and 2 points for returning the correct value. We did not award further partial credit.

B: We deducted 2 points for any errors with obtaining the payload_size_and_status field, 3 points for any errors with & with ~2 (or equivalent), 1 point for not returning, and 2 points for any header modifications. We did not award further partial credit.

C: We used a criteria of Correct (5) / Minor Error (3) / Major or 2+ Minor Errors (0). We did not award further partial credit. Here are some examples; more severe or multiple instances of these errors may have counted as a major error:

Minor:

- Doesn't shift status over by 1
- Doesn't save result back into header
- Assumes existing status value
- Incorrectly dereferenced header
- Only handles one status case (e.g. free or allocated)

D: We had several partial credit deductions, most importantly -2 for omitting a cast in pointer arithmetic, -2 for not adding the size of the payload, or -2 for incorrect handling of the last header on the heap.

E: We awarded 2 points for the edge cases where size is 0 or zone is NULL, 2 points for rounding up the requested size, 5 points for checking if the zone has enough available space to satisfy the request, 3 points for incrementing zone->ptr, and 3 points for returning the correct payload address. We awarded partial credit for the zone space check in the form of a minor error (-2) and major error (-5) deduction. We did not award further partial credit on this problem.

F: We had several partial credit deductions, most importantly -2 for incorrect pointer arithmetic, -3 for an incorrect cast or dereference, and -1 for not setting equal to next.

G: We had a deduction of -2 points per incorrect parameter and did not award further partial credit.

H: We had several partial credit deductions, most importantly -2 for incorrectly calculating payload start, -2 for an incorrect mathematical operation in calculating address difference, and -2 for a pointer type mismatch.