CS 240: Programming in C
Final Exam
Spring 2025

Practice Final Exam

Version I

Name: Solutions

Username:

Read all instructions before beginning the exam.

- This exam is intended to be equally or more difficult than the past official midterm provided.
- You are encouraged to post on Ed Discussion, without hesitation, any sort of question you may have about this practice exam.
- This is a closed book examination. No material other than those provided for you are allowed.
- You need only a pencil and eraser for this examination. If you use ink, use either black or blue ink. If you use pencil, your writing must be dark and clearly visible.
- This examination contains an amount of material that a well-prepared student should be able to complete in less than one hour.
- This examination is worth a total of 150 points. Not all questions are worth the same amount. Plan your time accordingly.
- Write legibly. You should try to adhere to the course code standard when writing your solution(s). Egregious violations may result in point deductions.
- Read each question carefully and only do what is specifically asked for in that problem.
- Assume appropriate includes have been added to the code segments shown in the problems.
- Circle your answer in true or false questions. On this exam problem 18 is false.
- Some problems require several steps. Show all your work. Partial credit can only be rewarded to work shown.
- Write your username on EVERY page where indicated. Any page without a username will receive a zero for the material on that page.

Signature:

Do not open the examination booklet until instructed.

1. (1 point) True or False: The `-S` GCC flag will generate an executable.

2. (2 points) Write a single valid to compile the C file named `purdue.c`, that includes `up.h`, with warnings as errors adhering to the C17 standard. The executable should be named `boilermaker`. Multiple valid answers.

```
gcc purdue.c -Werror -std=c17 -o boilermaker
```

3. (2 points) Which of the following correctly represents the stages of the compilation process using GCC, in order?

   [✓ A.] Preprocessing → Linking → Compilation → Object file creation
   B. Compilation → Object file creation → Linking → Preprocessing
   C. Preprocessing → Compilation → Object file creation → Linking
   D. Object file creation → Preprocessing → Linking → Compilation

4. (2 points) Write the conversion specifier to read a string composed of capital letters between T to L and digits from 0 to 9.

```
%[T-L0-9]
```

5. (1 point) True or [✓ False]: `fclose()` will not produce a segmentation violation if a `NULL` pointer is passed in as an argument.

6. (3 points) The following is the creation and allocation of a simple single linked list node with an integer value. Assuming successful memory allocation, would the code produce an error? If yes then state why this is the case, if no then state the output.

```
struct node *new = calloc(1, sizeof(struct node));
assert(new);
new->value = 240;
printf("%d\n", new->next);
```

```
0
```

7. (1 point) True or [✓ False]: The number of characters `printf()` will print depends on the size of the array of characters (string), and the `NUL` terminator contained by it.

8. (2 points) What is the value displayed by the `printf()` statement on a 64-bit system?

```
void read(char name[32]) {
    printf("sizeof = %lu\n", sizeof(name));
}
```

```
8
```

9. (3 points) Provide the `fscanf()` statement to read the name, `Linus Torvalds`, and the number, `1991`, from the following file input, ignore everything else. The file pointer is named `fp`, and the variables to store the inputs are called `name` (string with a buffer size of 100), and `num` (integer).

```
(Linus Torvalds)+(Linux:1991)
```

```
fscanf(fp, “(%99[^)])+(Linux:%d)”, name, &num);
```

10. (3 points) Implement a header guard for a header file using the macro identifier `UP_H`. Make sure to implement the preprocessor directives in order of appearance.

```
#ifndef UP_H
#define UP_H

#endif
```

11. (4 points) The following code segment has a signal handler that, when receiving a signal triggered by pressing `CTRL-C`, will cause the infinite while loop to end, terminating the program. However, when compiled at an unknown optimization level, pressing `CTRL-C` does not cause program termination as expected. Briefly explain what is causing the problem and propose a rewrite of a single existing non-blank line to fix the issue.

```
1.  unsigned char stop = 0;
2.
3.  void signal_handler(int x) {
4.      stop = 1;
5.  }
6.
7.  int main(void) {
8.      signal(SIGINT, signal_handler);
9.
10.     /* Pressing Ctrl-C invokes signal_handler() */
11.
12.     while (!stop);
13. }
```

```
Line 1

volatile unsigned char stop = 0;
```

12. (1 point) [✓ True] or False: `clearerr()` clears the end-of-file and error indicators for the stream pointed to by a file pointer.

13. (2 points) Which of the following can be used to create functionally equivalent behavior to `scanf()` when reading from `stdin`?

    A. `fwrite()`
    B. `sscanf()`
    C. `fprintf()`
    [✓ D.] `fscanf()`

14. (3 points) Given a structure with fields in this order: a structure pointer, a character pointer, and a long. What would be the size of the structure on a 32-bit system?

```
4 + 4 + 8 = 16
```

15. (3 points) Given a union with fields in this order: an integer, a structure pointer, and a character. What would be the size of the union on a 64-bit system?

```
8
```

16. (1 point) [✓ True] or False: The heap grows upwards, from lower to higher memory addresses.

17. (2 points) Use `typedef` to define a new type `node_t` as an alias for an existing structure `struct node`.

```
typedef struct node node_t;
```

18. (1 point) True or [✓ False]: In a pipelined food production process at McDonald's, it is essential to package fries before any other menu items to optimize throughput.

19. (3 points) What is the output of the following program? State the value displayed by `printf()`, undefined behavior, or error produced.

```
#define MOD(x) (-x)

int main() {
    int num = 10;
    printf("%d\n", MOD(num++));
}
```

-10

20. (3 points) Which of the following variables have their corresponding values stored in the stack?

```
struct node {
    int val;
    struct node *next;
};

int a = 10;

int main() {
    struct node *b = malloc(sizeof(struct node));
    static int c = 20;
    int *d = calloc(80, sizeof(int));
    char *e = "Bill Joy";
}
```

b, d, e

21. (3 points) What is the output of the following program? State the values displayed by `printf()`, undefined behavior, or error produced.

```
int x = 4;

void modifier() {
    int x = 1;
    x += 5;
    {
        extern int x;
        x *= 2;
    }
    printf("%d ", x);
}

int main() {
    modifier();
    printf("%d\n", x);
}
```

6 8

22. (2 points) What is the GDB command to resume the execution of the program until the next breakpoint or error?

c

23. (3 points) For the following code segment, what is the value displayed by the `printf()` statement on a little-endian system.

```
int main() {
    union data {
        unsigned long number;
        unsigned short bytes[4];
    } unit;

    unit.number = 0xA1B2C3D4E5F6A7B8;
    printf("%x %x\n", unit.bytes[1], unit.bytes[3]);
}
```

```
e5f6 a1b2
```

24. (4 points) Write the necessary single declaration for the bit flags in this order: `NONE`, `PASSPORT`, `BOARDING_PASS`, `I20`, and `I94`. There are multiple answers.

```
enum {                            enum {
  NONE = 0,                         NONE = 0,
  PASSPORT = 1,                     PASSPORT = 1 << 0,
  BOARDING_PASS = 2,                BOARDING_PASS = 1 << 1,
  I20 = 4,                          I20 = 1 << 2,
  I94 = 8,                          I94 = 1 << 3,
};                                };
```

```
#define NONE          (0)
#define PASSPORT      (1)
#define BOARDING_PASS (2)
#define I20           (4)
#define I94           (8)
```

25. (3 points) The following two code segments depict different implementations of a function. State which implementation is faster, and briefly explain why that is the case.

```
int first(int n) {
    if (n <= 1) {
        return 1;
    }
    return n * first(n - 1);
}

int second(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

First implementation is faster (first()).

Iterative solutions are faster than their recursive counterparts due to stack frame overhead.

26. (4 points) Create a function `convert_endian()`, that takes in as an argument and returns an `unsigned long`. Reverse the endianess of the argument and return it. Implement the function using casts. No credit will be given for other approaches.

```
unsigned long convert_endian(unsigned long num) {
    unsigned long r = 0;
    unsigned char *src = (unsigned char *)&num;
    unsigned char *dst = (unsigned char *)&r;
    for (int i = 0; i < sizeof(x); ++i) {
        dst[i] = src[sizeof(x) - 1 - i];
    }
    return r;
}
```

27. (2 points) Briefly explain what the `unsigned` keyword does.

Represents positive integers, i.e. not signed.

28. (1 point) True or [✓ False]: You can assign a pointer to an array of the same type.

29. (2 points) Given the following code segment, what is the value displayed by the `printf()` statement.

```
int array[] = { 2, 1, 6, 2, -3, 0 };
int *p = &array[2];
printf("%d\n", (p -= array[1] - array[3])[*(p - 2)]);
```

-3

30. (2 points) Based on the code segment in problem 30, what would the following `printf()` statement output in a 64-bit system.

```
printf("%lu %lu\n", sizeof(&array), sizeof(array));
```

```
8 24
```

31. (1 point) True or [✓ False]: `fwrite()` does not require the memory addresses of written variables.

32. (2 points) Write the function prototype of a function `func` that takes in two characters, and returns a pointer to a function that returns a void pointer and that takes no arguments.

```
void *(*func(char, char)) (void);
```

33. (2 points) Dynamically allocate a 2D array of integers, `array`, with 100 rows and 400 columns and initialize it to zero, in a single line.

```
int *array = calloc(100 * 400, sizeof(int));
```

34. (6 points) Define a recursive function `tree_height()`, which takes in a single parameter—a pointer to the root of a preexisting binary tree. The tree is made up of structures of type `tree_t`, with a `left` and `right` pointer to a node. This function returns the height of the tree (integer). A root node by itself has a height of 0.

```
int tree_height(tree_t *root) {
  if (root == NULL) {
    return -1;
  }

  int left_height = tree_height(root->left);
  int right_height = tree_height(root->right);

  return 1 + (left_height > right_height ?
    left_height : right_height);
}
```

For the following questions, assume that you have a doubly-linked list node structure of type `struct node` containing a single integer value, `val`, a `next` pointer, and a `prev` pointer.

35. (4 points) Write a function `remove_tail()` that takes the *address* of a pointer to the head of a doubly-linked list. Remove and deallocate the tail of the list, updating the relevant pointer(s). Assume the list is not empty.

```
void remove_tail(struct node **head) {
    struct node *cur = *head;

    for (; cur->next != NULL; cur = cur->next) {
    }

    if (cur->prev) {
        cur->prev->next = NULL;
    }
    else {
        *head = NULL;
    }

    free(cur);
    cur = NULL;
}
```

36. (4 points) Write a function `rewind_head()` that takes the *address* of a pointer to a node in the middle of a linked list. Make the argument point to the head of the list. This function returns void. Implement the solution *recursively*. Assume the list is not empty.

```
void rewind_head(struct node **node_ptr) {
    if (!(*node_ptr)->prev) {
        return;
    }

    rewind_head(&(*node_ptr)->prev);

    *node_ptr = (*node_ptr)->prev;
}
```

37. (2 points) Rewrite the second line in this code segment for it to produce a result without integer truncation.

```
int n = 10;
double result = n / 3;
```

```
double result = (double) n / 3;
```

38. (2 points) What statement about C macros is false?

    A. Macros are replaced by their values during the preprocessing stage.
    B. Macros can take arguments, similar to functions.
    [✓ C.] Macros are type-checked by the compiler.
    D. Macros can span multiple lines using the '/' character.

39. (1 point) [✓ True] or False: The `srandom()` function is used to seed the random number generator, and the `random()` function generates pseudo-random numbers based on that seed.

40. (2 points) Define a variable named `my_var` that is a pointer to an integer whose pointer cannot be modified.

```
int *const my_var;
```

41. (3 points) Write a function, `my_realloc()`, that when used in the following example will reallocate memory to the specified new size by the second argument, for the pointer and set it to `NULL` if reallocation fails.

```
int *ptr = malloc(10 * sizeof(int));
my_realloc(&ptr, 20 * sizeof(int));
```

Continuation of problem 41

```
void my_realloc(int **ptr, size_t new_size) {
    int *temp = realloc(*ptr, new_size);

    if (temp == NULL) {
        free(*ptr);
        *ptr = NULL;
    } else {
        *ptr = temp;
    }
}

/* realloc(3) is not required to solve this problem,
   but it is convenient. */
```

42. (1 point) [✓ True] or False: The `strcmp()` function stops comparing two strings when it encounters the `NUL` terminator in either of the strings.

43. (2 points) What is the conversion specifier for an `unsigned short`?

%hu

44. (5 points) Write a function named `toggle_bit()` that takes two arguments—an unsigned long and an integer—and returns the result of flipping (toggling) the bit in the first argument at the position specified by the second argument. The function must use bitwise operations to achieve this. No credit will be given for other approaches.

```
unsigned long toggle_bit(unsigned long num, int pos) {
    return num ^ (1UL << pos);
}

OR

unsigned long toggle_bit(unsigned long num, int pos) {
    return num ^ ((unsigned long)1 << pos);
}
```

45. (1 point) True or [✓ False]: A void pointer can be directly dereferenced without typecasting.

46. (2 points) Briefly explain a disadvantage of using global variables.

```
Poor maintainability as they can be accessed by any part of
the program.
```

47. (3 points) State the GCC command to compile `lookup.c` into `lookup.o` to be used later for building a shared library without linker errors.

```
gcc -c lookup.c -fPIC -o lookup.o
```

48. (3 points) In the parent directory (not in the library search path) there are two files, `lookup.a`, and `scanner.so`. State the GCC command to compile and link the C file in the current directory, `restric.c`, against the libraries, creating an executable named `section`.

```
gcc -o section restric.c -L.. -llookup -lscanner
```

49. (3 points) Write a macro named `ABS` that takes one argument and returns the absolute value of the argument.

```
#define ABS(x) ((x < 0) ? -(x) : (x))
```

50. (2 points) Which of the following optimizations is not typically included with the `-O3` flag?

A. Aggressive function inlining.
B. Vectorization of loops.
[✓ C.] Full debugging information generation.
D. Code motion to reduce runtime overhead.

51. (3 points) Given the following function and its corresponding stack dump, which line numbers contain the return address and integer `i`?

```
void dump(int x, int y) {
    int i = 0x11223344;
    long l = 0xbeefbeefdecafbad;
}
```

```
1.  0x7fffffffe3b0: 01    00    00    00    00    00    00    00
2.  0x7fffffffe3a8: 43    11    40    00    00    00    00    00
3.  0x7fffffffe3a0: b0    e3    ff    ff    ff    7f    00    00
4.  0x7fffffffe398: ad    fb    ca    de    ef    be    ef    be
5.  0x7fffffffe390: 00    00    00    00    44    33    22    11
6.  0x7fffffffe388: 2c    11    40    00    00    00    00    00
```

Return address: Line 2

Integer i: Line 5

52. (2 points) What does ASLR stand for?

```
Address space layout randomization
```

53. (2 points) Which of the following statements about system calls is true?

A. System calls allow a program to directly interact with the CPU registers.

B. System calls are executed entirely in user space without involving the kernel.

[✓ C.] System calls provide an interface for programs to request services from the operating system.

D. System calls are only available in assembly language and cannot be used in high-level languages.

54. (3 points) In hardware that supports double-buffering, state the prototype of the function that swaps video buffers before the next video blit or lock returns and after a vertical retrace occurs.

```
int SDL_Flip(SDL_Surface *screen);
```

55. (5 points) The following stack dumps represent a before and after of a string function. When finalized, the code generates an error. Many string functions could have been used to produce the stack dump, state two of them. Also, if the program terminates with an error, what would be the name of the mitigation the compiler used to detect it?

```
(Before)    0x7fffffffe3a4: 50    6f    65    00    00    45    a5    3f
            0x7fffffffe3ac: d4    ad    fd    b8    01    00    00    00
            0x7fffffffe3b4: 00    00    00    00    90    dd    d8    f7
            0x7fffffffe3bc: ff    7f    00    00    00    00    00    00

(After)     0x7fffffffe3a4: 50    6f    65    20    50    6f    73    74
            0x7fffffffe3ac: 20    6f    6e    20    45    64    20    66
            0x7fffffffe3b4: 6f    72    20    72    65    77    61    72
            0x7fffffffe3bc: 64    20    3b    29    00    00    00    00
```

strcpy() & strcat()

Stack canaries

56. (1 point) True or [✓ False]: GPIO stands for generic purpose in and out.

57. (2 points) Briefly describe an advantage of using `goto`.

Ease of readability when working with deeply nested code.

58. (6 points) Write a Makefile that compiles and links `magic.c` and `casting.c` into a static library named `conv`. Use implicit rules to create the object files.

```
conv.a: magic.o casting.o
    ar rcs $@ $^
```

59. (2 points) Briefly describe what `socket()` does.

Creates an endpoint for communication.

60. (2 points) What is the primary purpose of the function `SDL_LoadBMP()`?

   A. To initialize the SDL library for image handling.
   [✓ B.] To load a BMP image file into an `SDL_Surface`.
   C. To save an `SDL_Surface` as a BMP file on disk.
   D. To apply a color filter to a surface.

61. (1 point) [✓ True] or False: `sscanf` NUL terminates strings.

62. (1 point) We hope you enjoyed CS 240. Good luck in the final!

```
   ~~~~ ____   |~~~~~~~~~~~~~|
  Y_,___|[]|   | Boiler Up!  |
 {|_|_|_|PU|_,_|_____________|
//oo---00=00     000    000
```