# CS 240: Programming in C
# Final Exam
# Fall 2024

By Benjamin Lobos Lertpunyaroj

*Student-made Practice Exam 1*

**Name:** SOLUTIONS

**Username:**

## Read all instructions before beginning the exam.

- This exam is intended to be equally or more difficult than the practice final provided by the instructor.
- 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 approximately two hours.
- This examination is worth a total of 130 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.
- For true/false and multiple choice questions, simply circle your answer.
- Some problems require several steps. Show all your work. Partial credit can only be rewarded to work shown. The answer to question twenty-one is false.
- 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. (2 points) 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` with warnings as errors adhering to the ANSI standard. The executable should be named `boilermaker`. Multiple valid answers.

```
gcc -ansi -Werror -o boilermaker purdue.c
```

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. (2 points) True or (False): `fclose()` will **not** produce undefined behavior if a `NULL` pointer is passed in.

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);
```

```
No, 0    (ideal answer)

OR

Yes, due to the %d    (lenient checking required, not ideal).
```

7. (2 points) True or (False): The number of characters `printf()` will print depends on the size of the array of characters (string), and the `NULL` 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));
}
```

sizeof = 8

9. (2 points) Provide the `fscanf()` statement to read the name and the number 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).

```
(Jeffrey Turkstra)+(CS:252)
```

```
fscanf(fp, "(%99[^)])+(CS:%d)", name, &num);
```

Hard coding is allowed only for this question.

10. (2 points) What should always be done after freeing dynamically allocated memory?

Set pointer to NULL.

11. (2 points) Briefly describe the functionality of the header guard preprocessor directives.

Prevents redefinition errors during compilation.

12. (2 points) (True) or False: `clearerr()` clears the end-of-file and error indicators for the stream pointed to by a file pointer.

13. (2 points) Briefly describe a difference between declarations and definitions.

```
Declarations do not allocate memory (with some exceptions),
while definitions do.
```

14. (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()`

15. (2 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?

```
12 = 4 + 4 + 4
```

16. (2 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 (strictest requirement)
```

17. (2 points) (True) or False: The heap grows upwards, from lower to higher memory addresses.

18. (4 points) Create a structure declaration for a binary tree node that stores a character pointer, and an integer.

```
struct node {
  char *p;
  int val;
  struct node *left;
  struct node *right;
};
```

19. (3 points) Which of the following variables point to a section in the heap?

```
#include <stdio.h>
#include <malloc.h>

// valid structure declaration...

int main() {
    int a = 10;
    struct node *b = malloc(sizeof(struct node));
    static int c = 20;
    int *d = calloc(80, sizeof(int));
    char e[] = "C. K. May";

    // continues ...
}
```

b and d

20. (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;
```

21. (2 points) True or False: The C programming language was first implemented on an IBM System/360 mainframe by Dennis Ritchie.

22. (2 points) Briefly describe a use case for the `extern` keyword.

```
Declares a variable or function defined in another file.
```

23. (2 points) What is the gdb command to obtain a backtrace?

```
bt
```

24. (2 points) For the following code segment, what is the value displayed by the `printf()` statement on a little-endian system.

```
unsigned int x = 0x12345678;
unsigned char* p = (unsigned char*)&x;

printf("Result: %x\n", *(p + 1));
```

```
56
```

25. (4 points) Write the necessary single declaration for the bit flags in this order: `NONE`, `PASSPORT`, `VISA`, `I20`, and `I94`. There are multiple answers.

```
enum flags {
  NONE = 0,
  PASSPORT = 2,
  VISA = 4,
  I20 = 8,
  I94 = 16,
};
```

You can use 1 << n notation to solve this problem.

26. (2 points) Briefly explain why iterative solutions are faster than their recursive counterparts.

No stack frame overhead when doing things iteratively.

27. (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. Use bitwise operators. No credit will be given for other approaches.

```
unsigned long convert_endian(unsigned long num) {
  unsigned long out = 0;

  for (int i = 0; i < sizeof(unsigned long); i++) {
    out = (out << 8) | (num & 0xff);
    num = num >> 8;
  }

  return out;
}
```

Hard coding the bitwise for a unsigned long for this question is a valid solution.

Hard coding numerical values for bitwise, unless required for logic, is not allowed.

28. (2 points) Briefly explain what the `unsigned` keyword does.

Specifies that a variable can only have non-negative values.

29. (2 points) True or (False): You can assign a pointer to an array of the same type.

30. (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

31. (2 points) Based on the code segment in Question 30, what would the following `printf()` statement output in a 64-bit system.

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

8 24

32. (2 points) True or (False): `fwrite()` does not require the memory addresses of written variables.

33. (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);
```

34. (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));
```

35. (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);
}
```

Continuation of Question 35.

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.

36. (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;
}
```

37. (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;
}
```

38. (2 points) Rewrite the second line in this code segment to make it work without integer truncation.

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

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

39. (2 points) For the following code segment, will the string pointed to by `str` be stored in the heap? Why or why not?

```
int main() {
   char *str = "Hello, World";
}
```

No, it's pointing to .rodata.

If reasoning mentions that it is pointing to the stack, then only one point is given.

40. (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.

41. (2 points) 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.

42. (2 points) Define a variable named `my_var` that is a pointer to an integer whose pointer cannot be modified.

```
int *const my_var;
```

43. (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));
```

```
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.
```

44. (2 points) True or False: The `strcmp()` function stops comparing two strings when it encounters the `NULL` terminator in either of the strings.

45. (2 points) What is the conversion specifier for an `unsigned short`?

```
%hu
```

46. (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);
}
```

47. (2 points) True or (False): A void pointer can be directly dereferenced without typecasting.

48. (2 points) Briefly explain a disadvantage of using global variables.

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

49. (2 points) Given three object files: `myobj1.o`, `myobj2.o`, and `myobj3.o`, write the command to generate a library named "my lib" (the filename should be `libmy_lib.a` or `libmy_lib.so` depending on your approach).

```
ar rcs libmylib.a myobj1.o myobj2.o myobj3.o
```

50. (2 points) What are the two additional flags that must be passed to gcc to link against the library created in Question 49, assuming that it is located in the current working directory and that the current working directory is not in the library search path?

```
-L. -lmylib
```

51. (2 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))
```

52. (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.

53. (3 points) Given the following function and its corresponding stack dump, which line number contains 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 at line 2.

Integer at line 5.

54. (2 points) What does ASLR stand for?

Address space layout randomization.

55. (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.

56. (2 points) Briefly describe what `SDL_LockSurface()` is used for.

Lock the pixels of an SDL surface for direct access.

57. (5 points) The following stack dumps represent a before and after of a string function. When finalized, the code generates an error. Which was the string function used? If the program terminates with an error, what would be the name of the mitigation the compiler used to detect it? Multiple answers to the first question.

```
(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
```

strcat or strcpy

Stack canaries

58. (2 points) True or (False:) In C, reading and printing a Unicode string requires special handling.

59. (2 points) Briefly describe an advantage of using `goto`.

```
Ease of readability in deeply nested code.
```

60. (4 points) Write a Makefile that will compile a Latex document to `out.pdf`. Use a pattern rule to generate `.pdf` files by running `pdflatex` three times on the base name.

```
TARGET=out.pdf

all: $(TARGET)

%.pdf:
  pdflatex $*
  pdflatex $*
  pdflatex $*


There are various solutions.

Check out Lecture 25 for more information.
```

Continuation of Question 60.

61. (2 points) Briefly describe what `<ESC>[m` is and does.

ANSI escape code used to reset the text formatting attributes to default settings

62. (2 points) What is the primary purpose of the function `SDL_Flip()`?

    A. To initialize the SDL library for rendering.

    B. To flip a 2D texture horizontally or vertically.

    (C.) To update the entire screen with the contents of the back buffer.

    D. To apply a filter effect to a surface.

63. (2 points) (True) or False: `sscanf` `NULL` terminates strings.

64. (1 point) Find the clue hidden in this exam. Good luck on the final!