CS 240: Programming in C
Final Exam
Fall 2021

Name: Solutions

Username: | | | | | | | | |

Read all instructions before beginning the 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 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.
- You may leave after you have turned in all pages of the examination booklet. You will not be able to change any answers after turning in your examination booklet.
- 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.
- Do not attempt to look at other students' work. Keep your answers to yourself. Any violation will be considered academic dishonesty.
- Write your username on *EVERY* page where indicated. Any page without a username will receive a zero for the material on that page.
- The answer to question 63 is true.
- Read and sign the statement below. Wait for instructions to start the examination before continuing to the next page.

"I signify that the answers provided for this examination are my own and that I have not received any assistance from other students nor given any assistance to other students. Moreover, I will not discuss any part of this exam with anyone until after Saturday, December 18, 2021."

Signature:

- Do not open the examination booklet until instructed. Submission #: | | | | | | |

1. (2 points) True or False: Object files allow for incremental compilation, reducing the time it takes to compile "big" applications.

2. (2 points) Linking:

   A. Combines compiled object files together, producing an executable

   B. Produces machine code from source code

   C. Connects sockets together

   D. None of the above

3. (2 points) Write the gcc flag that turns warnings into errors.

   ```
   -Werror
   ```

4. (2 points) What is the line that needs to be added to the following code segment for it to successfully compile?

```
#include <stdio.h>

int main() {
  foo();
  return 0;
}

void foo() {
  printf("Hi\n");
}
```

   ```
   void foo(); /* before main() */
   ```

5. (2 points) True or False: Functions must be prototyped prior to their use, unless they have already been defined.

6. (2 points) Briefly describe the functionality of the #include preprocessor directive

   Includes the contents of a file into the source file during compilation.

   Most often this is a header file that is system or user defined.

7. (2 points) Provide the `fprintf()` statement to write the string, float to two decimal places, and integer to the file pointer specified by the argument.

```
void write_something(FILE *fp) {
  char str = "Hello!";
  float f = 3.1415;
  int i = 42;

  // fprintf() call

  return;
}
```

```
fprintf(fp, "%s %.2f %d", str, f, i);
```

8. (2 points) Why is it important to get the size argument correct when using `strncpy()`?

May cause unintentional omission of the null terminator if the size is inappropriate.

9. (2 points) What should always be done after closing a file pointer?

Set file pointer to NULL.

10. (2 points) Which of the following checks to see that a file can be opened with a specified mode:

    (A.) access()

    B. feof()

    C. ferror()

    D. clearerr()

    E. None of the above

11. (2 points) Write the conversion specifier to read a string composed of capital letters and lower case letters between c and k.

%[C-Kc-k] or %[A-Zc-k] depending on question interpretation.

12. (2 points) True or (False): You should use `assert()` to identify recoverable error conditions.

13. (2 points) Given a binary file full of integers, write the `fseek()` call that would move the file position to the 12th integer in the file.

```
fseek(fp, sizeof(int) * 11, SEEK_SET);
```

14. (4 points) Create a type and structure declaration for a doubly-linked list node that stores an integer. The type should be called "`list_node`".

```
typedef struct list_node {
  int val;
  struct list_node *next;
  struct list_node *prev;
} list_node;
```

15. (2 points) True or (False): A declaration allocates storage for a variable.

16. (2 points) Given a structure that stores two integers, write the compound literal that would assign the values 5 and 12 to the two fields in the variable below:

```
struct my_struct ms = { 0, 0 };

// Code to assign 5 and 12 to the two structure fields.
```

```
ms = (struct my_struct) {5, 12};
```

17. (2 points) True or (False): `extern` is used to define a variable.

18. (2 points) (True) or False: strings in C are arrays of characters.

19. (2 points) Given a structure with fields in this order: an integer, a character, and a short. What would the size of the structure be on a 64-bit system?

```
4 (int) + 1 (char) + 1 (pad) + 2 (short) = 8
```

20. (2 points) The call `fread(&my_data, 16, 2, fp);` will read how many bytes?

```
16 * 2 = 32
```

21. (2 points) True or (False): a file written by `fwrite()` on a big endian system can be read without modification using `fread()` on a little endian system.

22. (2 points) Briefly describe the difference between a structure and a union in C.

```
Structures allocate memory for each of its members separately.

Unions allocate memory to the size of its largest member.
```

23. (3 points) Declare an enumerated type named "`food`" that can take on the values: `CARROT`, `POTATO`, `TOMATO`, and `BREAD`.

```
typedef enum {
  CARROT,
  POTATO,
  TOMATO,
  BREAD,
} food;
```

24. (2 points) True or False: Big-endian systems have the most significant byte stored at the lowest address.

25. (4 points) Draw the truth table for the OR bitwise operator (|)

```
 x  y  |

 0  0  0
 1  0  1
 0  1  1
 1  1  1
```

26. (2 points) What is the value displayed by the following code:

```
int x = 8;
x = x << 1;
printf("%d\n", x);
```

```
8 * 2 = 16
```

27. (2 points) Briefly describe what a pointer is.

A variable that stores a memory address of another variable.

28. (3 points) Define a function that takes a pointer to an integer as an argument and returns nothing. The function should increment by one the value pointed to by the pointer.

```
void func(int *num) {
  (*num)++;
}

/* Assuming the pointer is not NULL */
```

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

30. (2 points) Given the following code, and assuming the first `printf()` displays "`arr = 8000`", what will the second `printf()` output?

```
int arr[100];
int *ptr = &arr[4];
ptr += 5;
printf("arr = %d\n", arr);
printf("ptr = %d\n", ptr);
```

ptr = 8036 8036 = 8000 + 9 * 4 (sizeof(int))

31. (2 points) Given the following code segment, what is the output?

```
int array[] = { 2, 4, 6, 2 };
int *ptr = &array[3];
printf("%d\n", *(ptr - *ptr));
```

4

32. (2 points) Describe the steps needed to compile a program and subsequently obtain a backtrace using gdb, assuming the program crashes.

```
$ gcc -g program.c
$ gdb ./a.out
(gdb) r
(gdb) bt
```

33. (2 points) Given a structure containing an integer x and the following code, write the more common syntax for accessing x using p.

```
struct my_struct s;
struct my_struct *p = &s;
int i = (*p).x;
```

```
p->x
```

34. (4 points) Write a code segment that uses `malloc()` to allocate a 40 element integer array, initializing each element to its index value.

```
int *arr = malloc(sizeof(int) * 40);
assert(arr);

for (int i = 0; i < 40; i++) {
  arr[i] = i;
}
```

35. (2 points) Why is it important to set a pointer to `NULL` after passing it as an argument to `free()`?

Avoid accessing a location in the heap no longer reserved (dangling pointers).

36. (2 points) What is wrong with the following code segment?

```
struct node *alloc_a_struct() {
  struct node my_node = { 0 };

  my_node.val = 42;
  return &my_node;
}
```

my_node is in the stack (local variable), and will get popped once the function returns, potentially causing garbage to be accessed from the returned pointer if the area was overwritten.

37. (3 points) Write a function, `my_free()`, that when used in the following example will free the associated memory and set the pointer to `NULL`.

```
int *ptr = malloc(sizeof(int));
my_free(&ptr);
// ptr should now be NULL
```

```
void my_free(int **ptr) {
  free(*ptr);
  *ptr = NULL;
}

/* Assuming malloc was successful and ptr is not NULL */
```

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

38. (4 points) Write a function named `remove_head()` that takes the *address* of a pointer to the head of a linked list. It should remove and return the head of the list, updating the relevant pointer(s). You may assume the list is not empty.

```
struct node *remove_head(struct node **head) {
  assert(head);
  assert(*head);

  struct node *temp = *head;
  *head = (*head)->next;
  temp->next = NULL;

  return temp;
}
```

39. (4 points) Given a pointer to the head of a singly-linked list (`head`) and an integer value `k`, write the code necessary to allocate and add a node to the beginning of the list.

```
struct node *new = malloc(sizeof(struct node));
assert(new);

new->val = k;
new->next = head;
```

40. (2 points) Define a pointer to a function that returns a pointer to a character and accepts two arguments, both single precision floating point values. The pointer should be named `the_func`.

```
char * (*the_func) (float, float) = NULL;
```

41. (2 points) True or (False) The following code will execute without error:

```
char *str = "Hello!";
str[2] = 'a';
```

memory allocated in .rodata not the stack (opposite is char str[]).

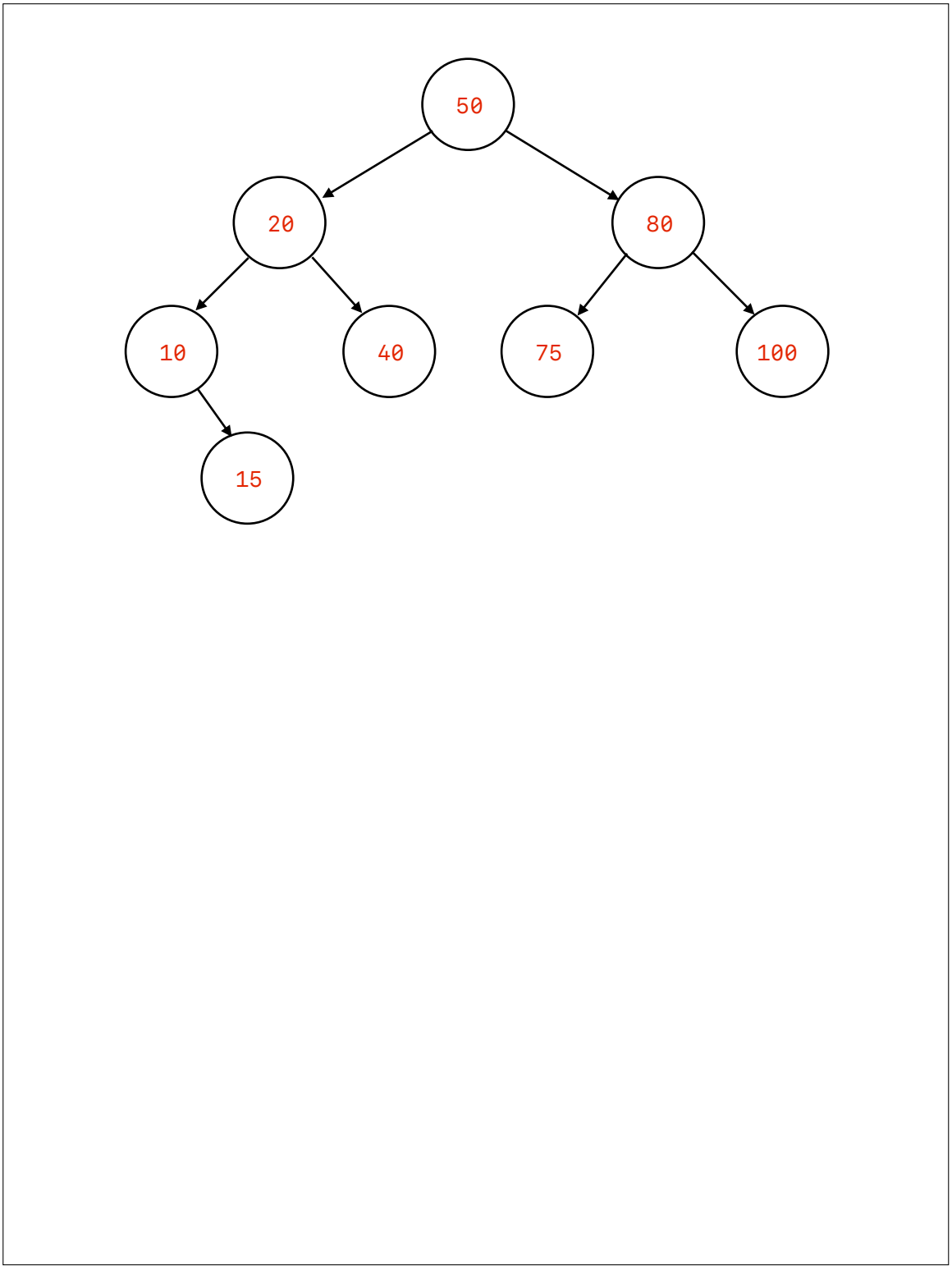
42. (3 points) Write a recursive function to calculate the nth number of the Fibonacci sequence (1, 1, 2, 3, 5, etc with each subsequent number obtained by summing the two previous).

```
int fib(int n) {
  if (n <= 1) {
    return 1;
  }

  return fib(n - 1) + fib(n - 2);
}

/* Assuming count starts from 0 */
```

43. (4 points) Write a recursive function that takes a single argument - a pointer to the root of a tree (type `struct tree_node`). Each node contains an integer value, `val`. Print the tree in postfix order.

```
void recursive_print(struct tree_node *root) {
  if (!root) {
    return;
  }

  recursive_print(root->left);
  recursive_print(root->right);
  printf("%d ", root->val);
}
```

44. (4 points) Draw the binary tree that results from inserting the following values in the order specified: 50, 20, 40, 80, 75, 10, 100, 15. Represent each node as a circle with the value inside.



45. (2 points) True or False: Iterative implementations of a given algorithm are typically faster than their recursive counterparts.

46. (2 points) Define a variable named `my_var` that is a pointer to an integer whose value cannot be modified.

```
const int *my_var = NULL;
```

47. (2 points) What does the `volatile` keyword mean?

Prevents the compiler from optimizing a variable under the assumption that it will be externally modified.

48. (4 points) Create a preprocessor macro that determines the maximum value of two variables.

```
#define MAX(a, b) (((a) > (b)) ? (a) : (b))
```

49. (2 points) What is one benefit of using a macro?

They avoid function call overhead, as they are expanded at compile time (preprocessor).

50. (2 points) The assignment below produces a compilation error. Rewrite the line so that it compiles.

```
char *c_ptr = NULL;
int *i_ptr = NULL;
c_ptr = i_ptr;
```

```
c_ptr = (char *)i_ptr;
```

51. (2 points) Why do we use a type of `void *` for parameters passed to callbacks?

Flexibility to pass a pointer of any type when using the callback.

52. (2 points) Why should you use `srandom()` prior to calling `random()`?

Allows for an initialization of the random number generator with a custom seed.

Traditionally random() will set the seed to 1 by default generating the same output.

53. (2 points) What is a variable of type `SDL_Surface` used for?

SDL_Surface is a struct that represents areas of "graphical memory", which can be drawn to.

54. (2 points) True or False: The `-O2` compiler flag tries to optimize while not producing a bigger executable.

55. (2 points) All of the following are approaches to improve runtime efficiency except:

    A. Using global variables if data is used more than twice in a function

    B. Using macros instead of short functions

    C. Using register variables

    D. Moving calculations outside of a loop when possible

"GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff"

56. (2 points) What is a system call?

Call to request a service from the OS's Kernel.

57. (2 points) Given the following function and stack dump, which line number contains the function's return address?

```
void hello(int value) {
  int local = 0xdecafbad;
}
```

Line 1: Nothing.
Line 2: Return address (in yellow).
Line 3: Stack frame pointer (saved base pointer (%rbp)).
Line 4: Local variable (in pink, little endian form).
Line 5: idk if someone can tell me thx lol

```
1. 0x7ffd11da7dc0: 00 00 00 00 00 00 00 00 ????????
2. 0x7ffd11da7db8: d3 12 40 00 00 00 00 00 ??@?????
3. 0x7ffd11da7db0: f0 7d da 11 fd 7f 00 00 ?}??????
4. 0x7ffd11da7da8: e0 51 36 8e ad fb ca de ?Q6?????
5. 0x7ffd11da7da0: e0 12 40 00 00 00 00 00 ??@?????
```

Line 2.

Return addresses point to low memory addresses (below global variables, stack and heap), to where functions are located (revisit diagram in lecture 12) (function are in .text, lower than .rodata, .bss, .data, etc).

Return addresses come before local variables, and as the stack grows downwards, this tells us that it has to be located in a higher memory address than line 4 (that holds the local variable).

58. (4 points) Name and briefly describe one mitigation that helps protect against buffer overflow attacks.

Stack canaries.

Known bytes that are inspected at the end of a function call, that if modified, causes the program to terminate, reporting potential stack smashing (SIGABRT).

(ASLR is also a valid answer).

59. (2 points) The steps for a client to connect to a server can include all of the following except:

A. socket()

B. bind()

C. listen()

D. connect()

E. All of the above

60. (2 points) Briefly describe the purpose of DNS.

To translate human-readable domain names into IP addresses.

61. (4 points) Write a function, `is_leaf()` that takes as an argument a pointer to the root of a binary tree and an integer value. Find the value in the tree (there will not be duplicates) and return a boolean true or false indicating whether or not it is a leaf node. Return false if the value is not in the tree.

```
bool is_leaf(tree_t *root, int val) {
  if (!root) {
    return false;
  }

  if (root->val == val) {
    return ((root->left == NULL) && (root->right == NULL));
  }

  return is_leaf(root->left) + is_leaf(root->right);
}
```

62. (2 points) Given the following Makefile, what is the purpose of `$^`?

```
CC=gcc
CFLAGS=-I.
DEPS = hellomake.h
OBJ=hello.o hellofunc.o

%.o: %.c $(DEPS)
  $(CC) -c -o $@ $< $(CFLAGS)

hello: $(OBJ)
  gcc -o $@ $^ $(CFLAGS)
```

To expand the list of prereqs for 'hello' (target), which corresponds to 'hello.o hellofunc.o' ($(OBJ)).

63. (2 points) (True) or False: A toad is a frog.