CS 240: Programming in C
Midterm Exam 1
Spring 2024

Name:

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 well under one hour.
- This examination is worth a total of 100 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.
- 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.
- 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."

Signature:

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

1. (25 points) Write short answers to the following questions.

   (a) (2 points) Write a single, valid command with which you would use gcc to compile the C file named `boiler.c` with warnings enabled using the C99 standard. The executable should be named up. There are multiple valid answers.

   (b) (2 points) Describe briefly what the `-c` compiler flag does and why one would use it.

(c) (2 points) True or False: `scanf()` always NUL terminates strings.

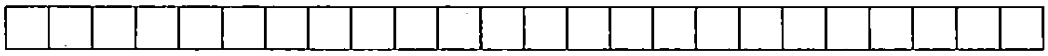
(d) (2 points) Why must the arguments corresponding to conversion specifiers in `scanf()` and related functions be pointers?

(e) (2 points) What is wrong with the code below? How might you fix it?

```
char buf[8] = { 0 };
char str[] = "Hello, my name is Joe";
strncpy(buf, str, 8);
```

(f) (4 points) Given a structure containing the following members in the provided order, indicate the layout in memory using the boxes below. Each box corresponds to one byte. Designate each variable's location(s) by writing its letter in the box(es). Any padding should be indicated with a 'P'. Assume a 64-bit architecture.

```
long l;
char c;
short s;
```



Given the following code segment and a 64-bit architecture, answer questions 1.g. and 1.h.

```
short array[8] = { 1, -1, 4, -2, 1·};
short *ptr = NULL;
ptr = &(array[2]);
printf("%hd\n", ptr[*(ptr + *ptr + *(ptr + 1))]);
```

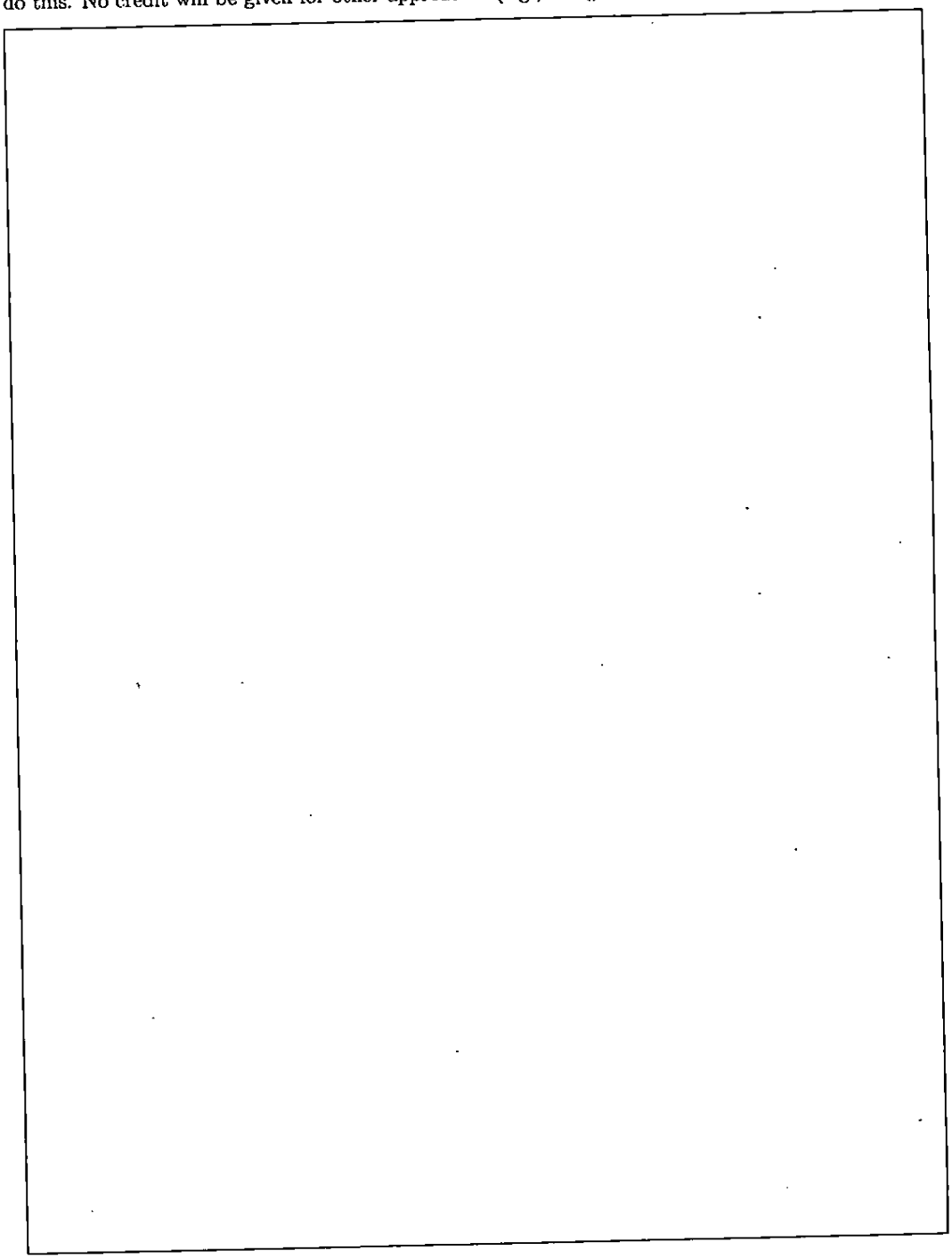
(g) (2 points) What is the value of *(ptr + 2)?

(h) (2 points) What is the value displayed by the `printf()` statement?

(i) (2 points) The code fragment below contains two code standard violations. Identify and describe one of them.

```
for (int k=0; k < num_iters; k++) {
   printf("Spring is on its way!");
}
```

(j) (5 points) Write a function named `bit_enabled()` that takes two arguments—an `unsigned long` and an integer—and returns a boolean value indicating whether or not the bit in the first argument at the position specified by the second argument is set (is 1). You must use bitwise operations to do this. No credit will be given for other approaches (e.g., using modulus, division, etc).



2. (15 points) Answer the following questions.

   (a) (4 points) Declare an enumerated type named `tree_species` that can take on one of the following values: `BLUE_SPRUCE`, `BALSAM_FIR`, `SCOTCH_PINE`, and `RED_CEDAR`. Simultaneously create a type named `species_t` that refers to it.

(b) (4 points) Declare a structure, `tree_struct`, that contains—in this order—a string of `MAX_LEN` (=32) elements named `address`, a `species_t` member named `species`, a single-precision floating point value named `radius`, a long integer value named `height`, a single-precision floating-point value named `weight`, a single-precision floating-point value named `dry_weight`, and an integer named `purchased`.

(c) (2 points) Use `typedef` to declare a new type, `tree_t`, of the struct in 2.b.

(d) (2 points) Given the following: `sizeof(double) = 8`, `sizeof(float) = 4`, `sizeof(char) = 1`.

```
tree_t trees[20];
printf("sizeof = %ld\n", sizeof(trees));
```

What would be printed to the screen?

(e) (3 points) Write a function named `tree_moisture()` that accepts one parameter of type `tree_t`. Assume all members of the passed in structure are populated. The function should return the tree's moisture content, a single-precision floating point value.

A tree's moisture content is determined by the following equation:

$$Mc = \frac{Wg - Wo}{Wo} * 100 \tag{1}$$

Where Mc is moisture content, Wg is green weight, and Wo is oven dry weight.

You should use the weights in the provided structure for Wg and Wo.

3. (40 points) Below about 50% foliar moisture content, Christmas trees ignite readily from matches, and at less than 20%, they burn with great violence. Write a function named `nero_number` that determines and returns the number of houses that are likely to catch fire before Christmas for a set of data formatted per below. It should accept two arguments (in this order): the input filename (a string) and an output filename (a string).

   Additionally, this function should produce an output file for the fire department that lists the address and first day of probable ignition (the day the tree reaches 20% or less moisture content) for each house that ignites.

   The input file should be in the following format:

   `address,species,radius,height,weight,dry_weight,day_purchased`

   `address` is a string containing any character except the delimeter.

   `species` is an enumerated type and as such treated as an integer for the conversion specifier.

   `height` is the height in feet and inches. E.g., 5'2". You may assume no tree is shorter than one foot. The trailing inches value is optional. I.e., you could have a tree that is 6'.

   `radius` is the radius of the base of the tree in inches. E.g., 42".

   `weight` is the tree's green weight.

   `dry_weight` is the tree's dry weight.

   `day_purchased` is the day of the year (numbered from 1 to 365) the tree was purchased. Note that Christmas occurs on day 359.

   Moisture content reduction of a tree varies by species and is given by the table below:

   Table 1: Foliar Drying Rates

Species	Drying Rate per Day
BLUE_SPRUCE	6.2%
BALSAM_FIR	6.4%
SCOTCH_PINE	4.1%
RED_CEDAR	5.3%

   For a house to catch fire, the purchased tree must have attained a moisture content <= 20% on or before Christmas Day (day 359) AND have a volume larger than 100,000 cubic inches.

   For our purposes, we will assume the tree is a perfect cone. The volume of a cone can be computed as follows:

$$V = \pi * r^2 * \frac{h}{3} \tag{2}$$

   Where π can be approximated as 3.14159, r is the radius, and h is the height (both in inches).

   A sample input file follows:

```
1838 Dykhouse Ave;1;46";5'2";30.2lbs;20.5lbs;300
547 Jacob Ave;3;36";5'8";22lbs;18lbs;320
15090 178th;2;38.2";6';35lbs;22.3lbs;310
27 Hilltop Dr Apt 9;1;40";5'3";25lbs;10lbs;315
```

   Note that each field is separated by a semi-colon (;) and that, for the height, the inches portion is optional.

You may wish to use the earlier structure declaration to hold these values, but that is not required. You may also assume that functions defined earlier in the exam are available for use without redefining them.

The output file should contain the address followed by a colon and space, followed by the day on which the tree is likely to combust.

DO NOT assume that data in the input file are without error. Strings that are not able to be stored in `MAX_LEN` (=32) bytes should be considered to be one of the possible errors. Also, ensure that the files have been properly opened. If ANY error occurs, return -1. Do not forget to set the file pointers back to `NULL`.

A sample output file corresponding to the above input file follows:

```
15090 178th: 335
27 Hilltop Dr Apt 9: 345
```

The return value would be: 2

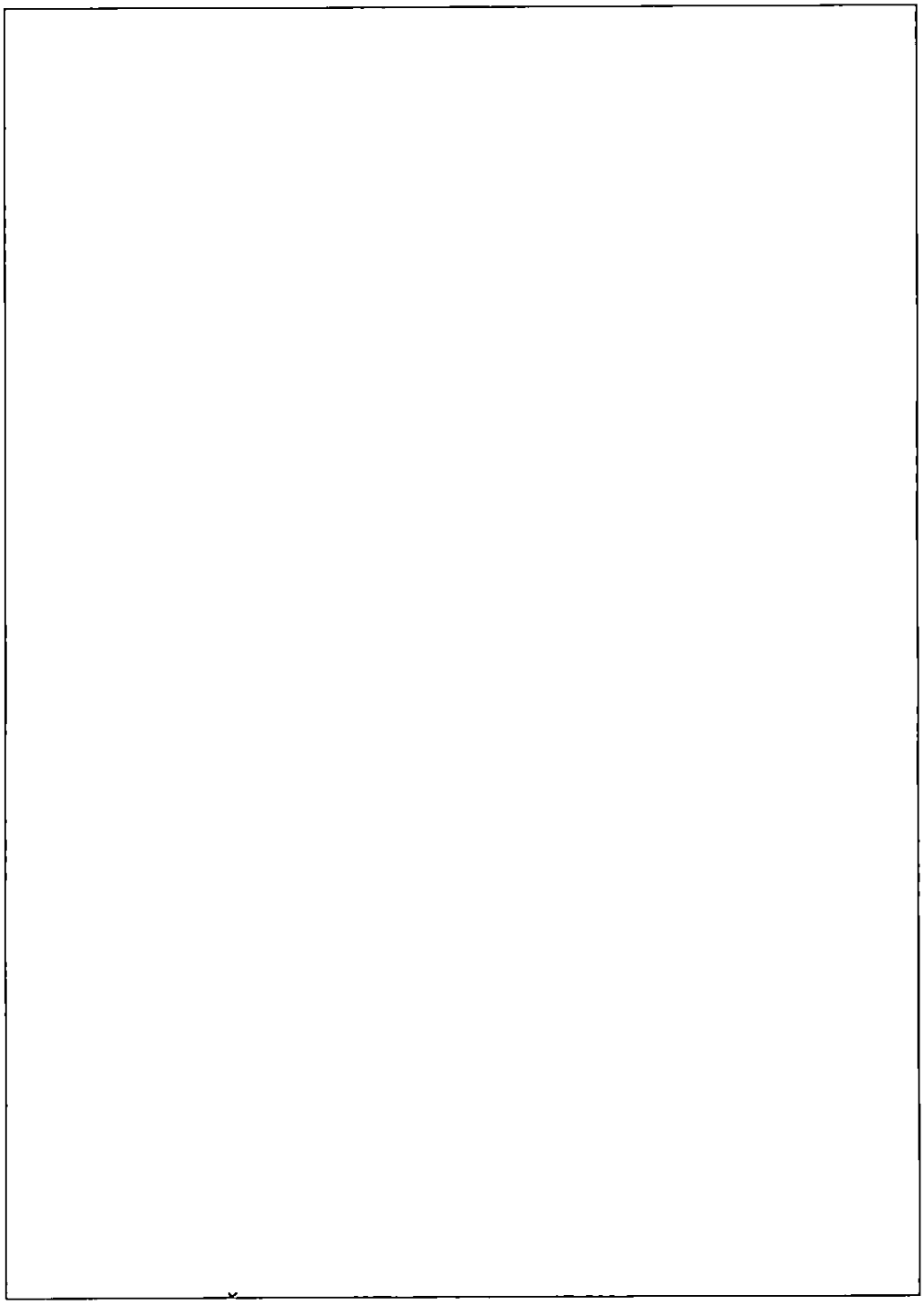
Be sure to include appropriate assertion checks for the function arguments. You should provide the appropriate `#includes` as well.

Follow the code standard as much as possible, but do not spend too much time on comments.

Hint: Pay close attention to `fscanf()`'s return value. Use `ftell()` and `fseek()` to attempt to re-read a record if it does not conform to the first format.

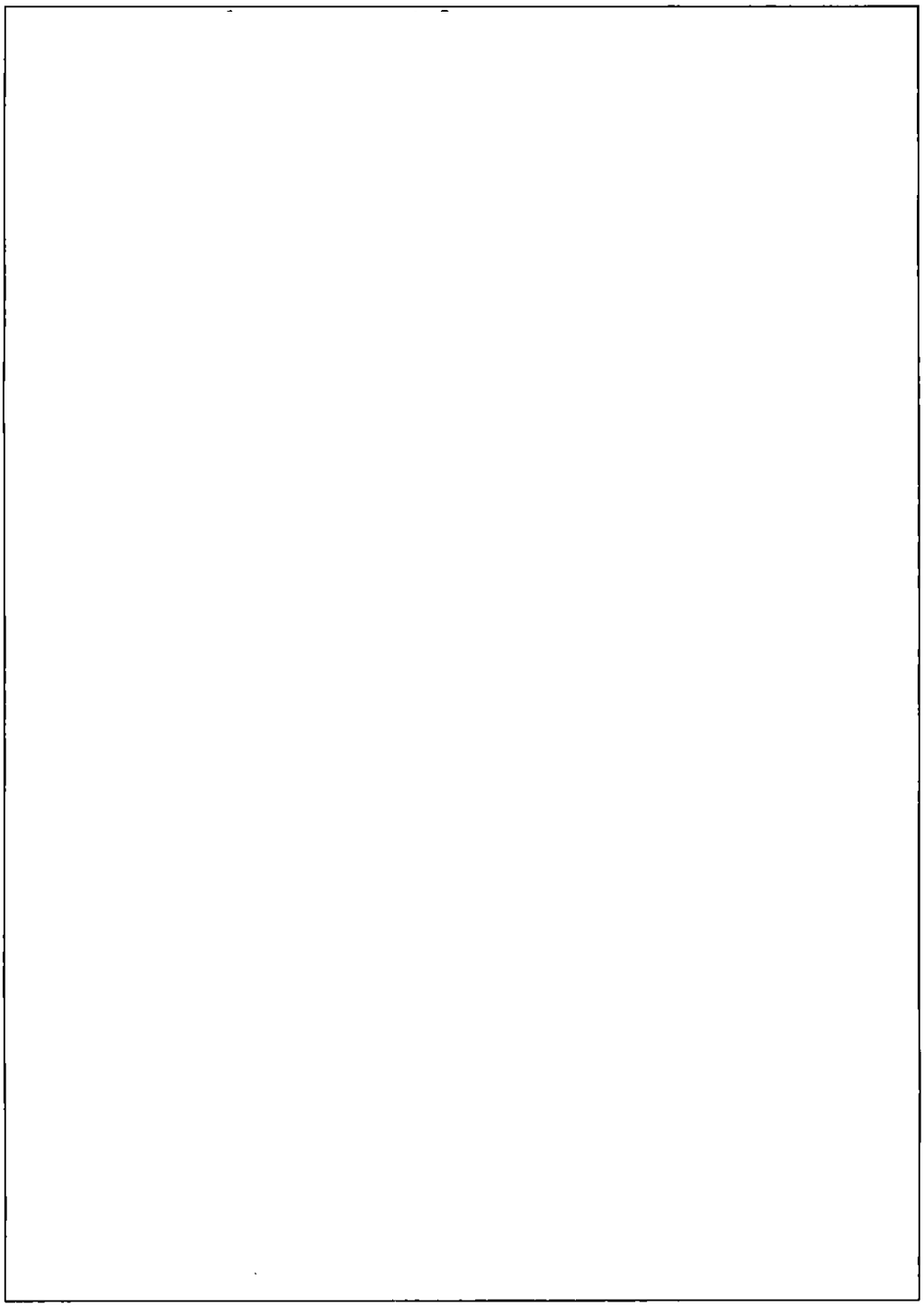
Do not write code on this page, it will not be graded.

Work area for problem 3...



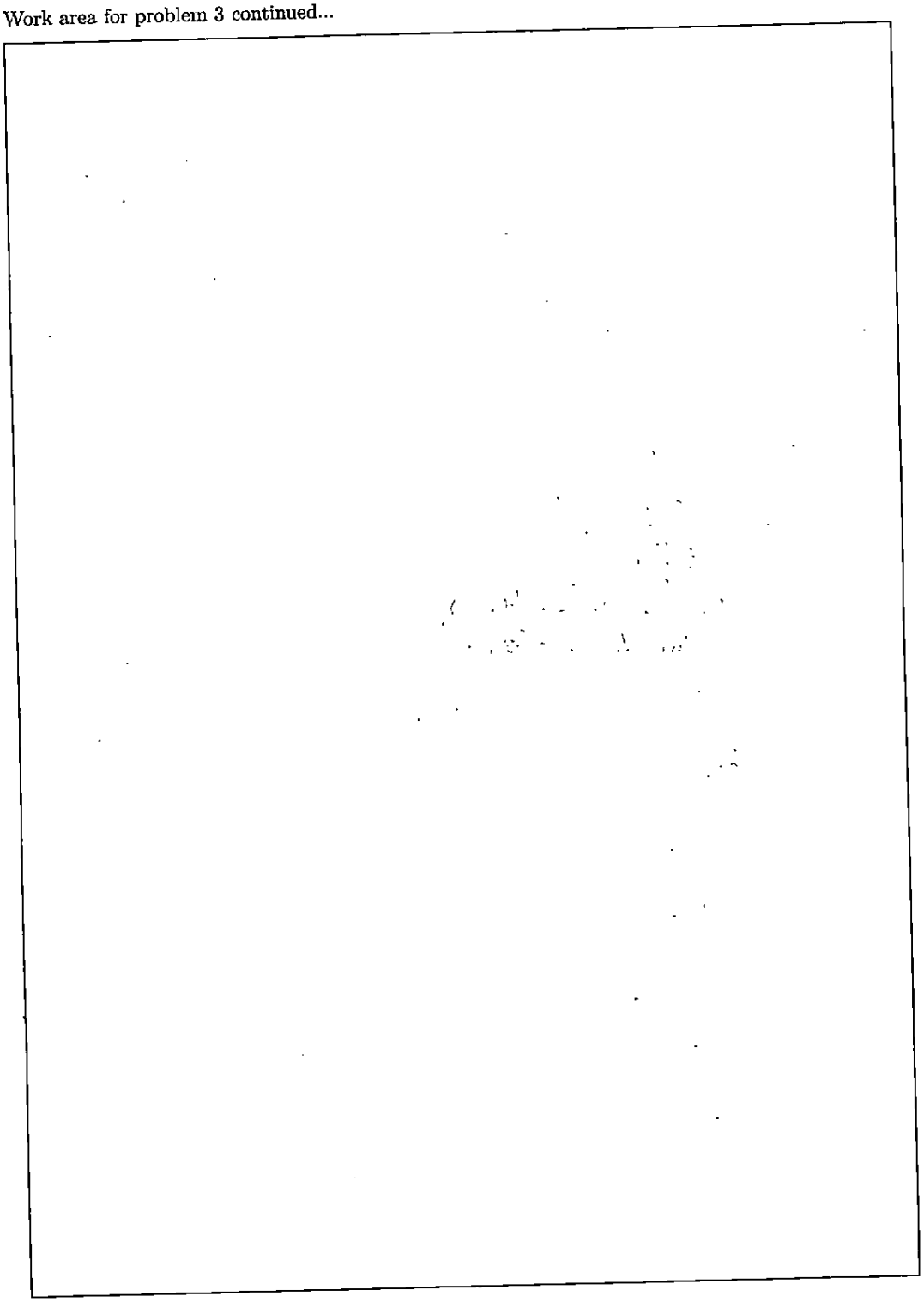
Turn over for additional space...

Work area for problem 3 continued...



Turn over for additional space...

Work area for problem 3 continued...



4. (20 points) Write a function, `reverse_endian()`, that accepts one argument—a file pointer that has already been opened for read and write. The function should return a `long` that is the largest number encountered while processing the file.

   The file pointer refers to a binary file containing sequences of variable length integers. The variable length integers are represented by a multi-byte value where the first byte indicates the number of bytes that follow and may range between 1 and 8, inclusive. The bytes that follow contain the value of the integer.

   Convert the integers, in place, from little endian to big endian and return the largest number encountered in the file. Assume that the architecture on which the program runs is little endian.

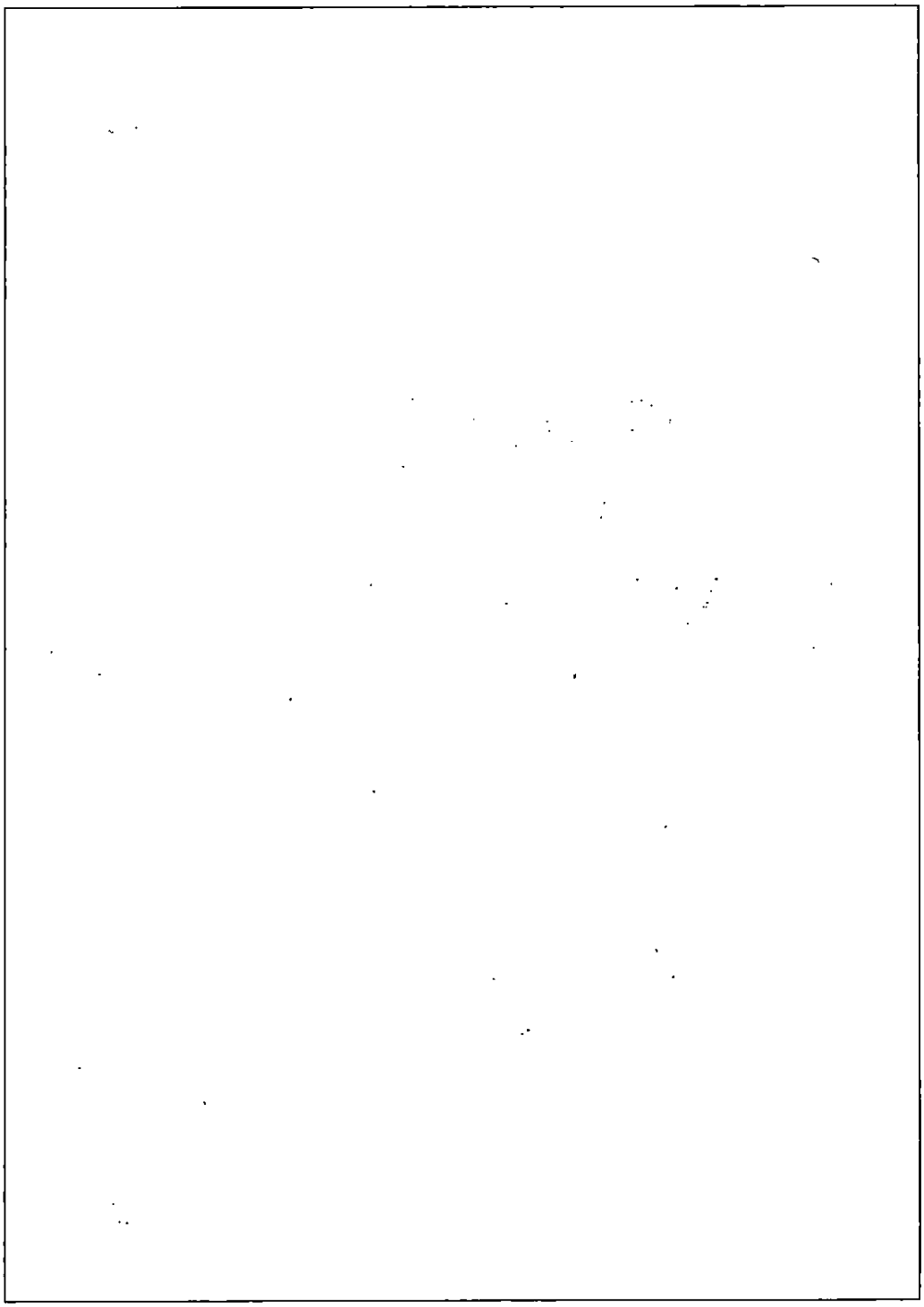
   Include assertion checks for the argument. Assume no other errors can occur.

   A sample input file (dumped using xxd) follows...

```
00000000: 03fa 9e5d 0495 c2c3 8504 779d 4d8b 039f  ...]......w.M...
00000010: cc0b 042d 6415 5308 84bc 5bcc dfd0 856a  ...-d.S...[....j
00000020: 0423 c74e b803 113d ad01 db04 1455 9998  .#.N...=.....U..
```

Turn over for additional space...

Work area for problem 4 continued...



Work area for problem 4 continued...