**Please note that this is not a complete midterm.**

## Problem 1

For each of the following fragments of C code,

- If there's a (potential) bug, write **bug**; if it's bad style write **style**. (Or write both, or none!)
- Explain *briefly* what the bug or style problem is, if there is one.
- Show a better way to write it.

1.
```
log_s = 0;
while (s = s/2)
        ++log_s;
```

2.
```
#define TRUE 1
#define FALSE 0

b = f(x);
if (b==TRUE)
        return;
assert(b==FALSE);
```

3.
```
#define N 100
#define INRANGE(x) (0 <= (x) && (x) < N)

while (INRANGE(a[++i]))
        printf("%d\n",a[i]);
```

## Problem 2

Translate the English-language description into a C declaration, and the C declaration to an English-language description.

1. Procedure *p* (returning nothing) taking one *Tree* argument using call-by-reference, where Tree is a pointer to struct containing string and two subtrees.
2. `double (*E)[8];`

## Problem 3

Put the appropriate assertion *before* each of these code fragments. Write auxiliary functions as necessary to implement your assertions.

Assume that an appropriate assertion:

1. Guarantees against doing something bad (accessing memory out of bounds, infinite loop, etc.);
2. Subject to item 1, is as permissive as possible.

Write your assertions in English if you can't figure out how to write them in C.

a. Given `struct {int a; char *s;} vec[30], *p;`

```
p = vec;
p += N*8;
p->a = vec[M].s[0];
```

b.

```
    q->next = p->next;
    free(p);
    if (q->next == NULL)
      a=0;
```

## Problem 4

Write a function `void putoct(unsigned n, int w)` that prints the value of its given argument `n` in octal, result right-justified in a field of `w` spaces. If `n` won't fit in `w` spaces, or if `w` $< 0$, `n` is simply printed. The output always has exactly one leading `0`. The table below shows `putoct`'s output for various values of `n` and `w`; "␣" denotes a space. Use more space on the next page if necessary.

| n | w | output |
|---:|---:|---|
| 0 | 3 | ␣␣0 |
| 65535 | 4 | 0177777 |
| 968 | -1 | 01710 |
| 4294967295 | 20 | ␣␣␣␣␣␣␣␣037777777777 |

## Problem 5

Write a C function, `int *flatten(struct node *list, int *len)`, that returns an array that contains a flattened version of the linked list and sets `len` to be the length of the new array. That is, the array contains the integer values in the elements of the list, in the order in which they appear in the list. Assume `struct node` has the following definition:

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

You may write auxilliary functions as needed.

## Problem 6

The following is a description of a CS217 assignment from a previous year. Your task is to sketch out a design for this program. Your design should specify the necessary modules. For each module, you should specify its purpose, the data structure(s) provided, and the routines provided. You do not need to write code for the routines, just specify what they should do.

*Write a C program that reads text from standard input and produces paragraphs of left- and right-justified text on standard output. The input consists of zero or more lines of zero or more characters. The input is separated into paragraphs by one or more blank lines (i.e., lines consisting solely of whitespace).*

*The output consists of paragraphs that are 79 characters wide (not including the terminating newline character). The first line of a paragraph is indented 5 spaces. Paragraphs are not separated by blank lines. All lines, except for the last line of a paragraph, are right justified. There can be a single space between a period and the next word.*

*Your program should insert extra spaces between the words of a line to achieve both left- and right-justification. These spaces should be evenly distributed.*