Name:	COS 217 Midterm
	Fall 2008
	October 23, 2008

Pledge:

Directions:

- Please answer each question in the space provided. The amount of space should be sufficient for a correct answer. If you need more space, please use the backs of pages, and make a note to that effect. If you run out of space, exam books are provided at the front of the room.
- This exam is open-book, open-notes, and is covered by the Honor Code. Please write and sign the pledge after you finish your exam.
- There are a total of four sections, with the number of points for each shown by the question. While it is not the intent for the exam to be a race, spending too much time on a single question may preclude finishing the exam. Budget your time wisely.
- To be fair, I will try to avoid answering content-related questions during the exam, unless it's to correct a mistake on my part.
- If you feel that a question **requires** additional assumptions or information to answer, please state them. Your guiding principle should be *Occam's razor*, which loosely translated states that you should allow as few assumptions as necessary to explain the situation.
- Answers should assume C/Unix and a compile environment like our settings on hats unless otherwise stated or implied
- Please first read over the entire exam and then begin to answer questions. I will be available in the hall way in the event that you have questions. Please do not loudly ask questions in the exam room itself.
- Please write legibly

#	Name	Points Available	Score
1	True or False	10	
2	Short answer	35	
3	Code read/write	30	
4	Bug hunt	25	
	Total	100	

- 1. True or False (10pts) For each statement, write "true" if the statement evaluates to true, or "false" if the statement evaluates false. If you believe the statement does not have a clear answer, give whichever choice is more appropriate and explain why.
 - 1 && 2 && 3 && 4

• 10 > x > 5

• 0x100 - 64

• -5 >> 2

• (5 < 10) ? 0 : 1

- 2. Short Answer (35pts) Answer each item well in no more than 3 sentences.
 - What is a dangling pointer? Give two example code sequences showing how it can arise.

• What is the output of the following code sequence:

• Assuming x is valid, what would you put in the body of the function in order to change the field y to 5?

```
struct blah {int y;};
typedef struct blah *blah;
static void Func(blah *x)
{
}
```

 \bullet What does the following code segment do:

```
void *p = malloc(1);
*p = 1;
```

• What is the difference between the stack and the heap?

• Write two function declarations as follows: the first should accept a character buffer and maximum length from the caller, and return the number of bytes filled. The second should accept a maximum length from the caller, returning a buffer it allocates, as well as the number of bytes filled. Use only basic data types, not structures.

• Convert the body of Func to use a for loop (not while/do):

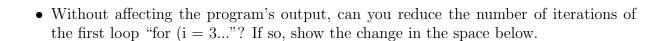
```
static void Func(int i)
{
    do {
       Func2();
       i--;
    } while (i > 3);
}
```

3. Code reading and writing (30pts)
Consider the following piece of code:

```
#include <stdio.h>
enum {MAX_COUNT = 5};
#ifndef TRUE
#define TRUE 1
#endif
#ifndef FALSE
#define FALSE 0
#endif
static int vals[MAX_COUNT];
static int numFound = 0;
int main(int argc, char *argv[])
  int i;
  int j;
  vals[numFound++] = 2;
  for (i = 3; numFound < MAX_COUNT; i++) {</pre>
    int match = TRUE;
    for (j = 0; j < numFound; j++) {
      if ((i % vals[j]) == 0)
        match = FALSE;
    }
    if (match)
      vals[numFound++] = i;
  }
  for (i = 0; i < numFound; i++)</pre>
    printf("%d\n", vals[i]);
  return(0);
```

•	What is the value of numFound immediately before the first loop?
•	What is the final value of numFound?

• What is the output of this program?



• Without affecting the program's output, can you reduce the number of iterations of the second loop "for (j = 0..."? If so, show the change in the space below.

4. Bug hunt! (25pts)

The following code is supposed to read lines from the user, store them in a list, and write the lines back from the last line to the first. Each line is guaranteed to be no more than 99 characters long. However, it's full of bugs. Identify **and briefly explain** as many bugs, possible bugs, and design flaws as you can. Fix the program so that it works and is good stylistically. Five correctly-executed changes receives full credit.

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
enum \{MAX\_LINE = 99\};
typedef struct LineBuf {
  char lb_line[MAX_LINE];
  struct LineBuf *lb_next;
} LineBuf;
static LineBuf *head = NULL;
int main(int argc, char *argv[])
{
  char line[MAX_LINE];
  LineBuf *walk;
  while (scanf("%d", line) == 1) {
    LineBuf *lb = malloc(sizeof(LineBuf *));
    assert(lb != NULL);
    lb->lb_line = line;
    lb->lb_next = head;
  }
  for (walk = head; walk != NULL; walk++)
    printf("%s\n", walk->lb_line);
  return(0);
}
```

this page left intentionally blank to answer the previous question