COS 217 Midterm

Fall 2006

Please write your answers clearly in the space provided. For partial credit, show all work. State all assumptions. You have exactly 50 minutes for this exam. This midterm is open book, open notes. Put your name on every page. Write out and sign the Honor Code pledge just before turning in the test. "*I pledge my honor that I have not violated the Honor Code during this examination.*"

Question	Score
1	/30
2	/15
3	/15
4	/40
Total	/100

Name:

Honor Code:

1 Short Answer

Answer the following in only the space provided.

1. What is the difference between a dangling pointer and leaked memory?

2. What is the difference between an ADT and a data structure?

3. Give two reasons to have virtual memory?

4. Write C code that puts the string "Heap" into the heap segment.

5. Write C code that puts the string "Stack" into the stack segment.

6. After the statement `unsigned char x = 0121;`, what is the value of `x` interpreted as an 8-bit signed integer value (expressed in decimal)?

7. After the statement `unsigned char x = 0xF0;`, what is the value of `x` interpreted as an 8-bit signed integer value (expressed in decimal)?

8. Describe one property a good hash function should have. Why should it have this property?

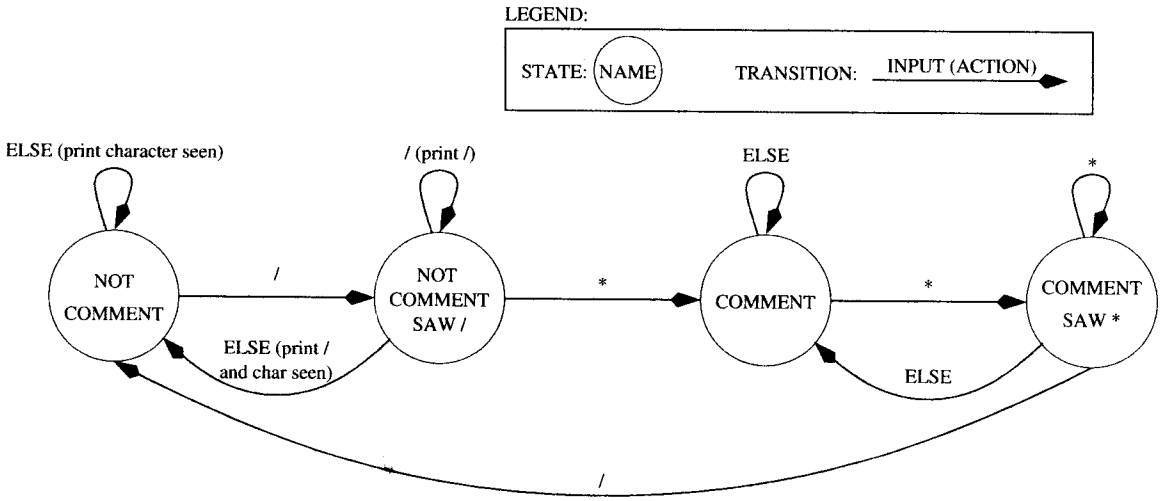
2 Sign Extension

Prove that sign extending the 3-bit two's-complement number $a_2a_1a_0$ to the 4 bit two's-complement number $b_3b_2b_1b_0$ preserves its value.

3 DFA

Draw a DFA with actions to print only the content of string literals found in C code (not in comments and without interpreting control sequences). Please clearly indicate your start state. Things to consider include:

1. While the "-character may exist inside comments, strings do not exist inside comments. In other words, no part of a comment should be printed.
2. Just as strings cannot start in comments, comments cannot start inside strings.
3. Valid strings may include the control sequence \" as part of the string. Do not interpret this control sequence. (Interpreting it would just print a ". Your DFA should print both the \ and " characters in sequence.)
4. Comments start with `/*` and may have many `/*`'s inside of them.
5. Comments end with `*/` but may not have any `*/`'s inside of them.
6. If you find the following figure useful, you may use it as part of a larger DFA which constitutes your answer. Note: you may need to modify, not just add to this figure. This figure is not guaranteed to perform any specific task, and it does not have a start state.



4 Bug Hunt!

The following code on the next page attempts to implement a very simple editor. The pages following that contains man pages for your convenience. Find eight *distinct* bugs (there are more than eight total bugs in the program) using the guide below. Keep in mind that some lines may participate in more than one bug. Please list line numbers when describing each bug. Write your answer to this part on this page only. Note that the code on the next page compiles (with a warning) and executes, so don't worry about finding syntax errors.

1. The compiler (gcc with no arguments other than the C file name) prints "warning: initialization from incompatible pointer type" as its only message due to a bug on one line in the function `main`. What is the line number and what is the problem?

2. Describe a bug related to variable scope in the `insert_text` function which may lead to a segmentation fault at run time for some inputs.

3. Describe a bug which may lead to a segmentation fault at run time for some inputs.

4. Describe another bug which may lead to a segmentation fault at run time for some inputs.

5. Which paragraph of code makes printing and exiting needlessly less convenient for the user?

6. Describe another bug or a serious style problem.

7. Describe another bug or a serious style problem.

```
0   #include <stdlib.h>
1
2   /* Insert text at pos in buf, shifting existing text to make room */
3   int insert_text(char *buf, int pos, char *text)  {
4     int size, i;
5
6     /* Allocate memory for the new text */
7     size = strlen(text);
8     buf = (char *)realloc((void *)buf, size);
9
10    /* Insert the text and shift text in buffer to make room */
11    for(i = 0; i < size; i++) {
12      buf[i+pos+size] = buf[i+pos];
13      buf[i+pos] = text[i];
14    }
15
16    return pos + size;
17  }
18
19  /* Find pattern in buf and return its staring position */
20  int search(char *buf, int pos, char *pattern) {
21    char *match;
22
23    /* Search for string pattern, return end position if not found */
24    if(!(match = strstr(buf, pattern)))
25      return strlen(buf);
26
27    return match - buf;
28  }
29
30  int print(char *buf, int pos, char *junk) {
31    /* Print the buffer */
32    printf("The editor's buffer contains:\n");
33    printf(buf);
34  }
35
36  int main(int argc, char *argv[]) {
37    int (*fn[5])(char*, int, char*) = {exit, insert_text, search, print};
38    int command, text_pos, pos = 0;
39    char *buf, ch, text[100];
40
41    /* Create some space for the initial buffer to get things going */
42    buf = (char *) malloc(1);
43
44    while (1) {
45      /* Give menu of commands and get command */
46      printf("0:Exit, 1:Insert at Position %d, ", pos);
47      printf("2:Search for String, 3:Print Buffer \n > ");
48      scanf("%d\n", &command);
49
50      /* Get text for command and execute */
51      text_pos=0;
52      while((ch = getchar()) != '\n')
53        text[text_pos++] = ch;
54      text[text_pos] = '\0';
55
56      pos = (*fn[command])(buf, pos, text);
57    }
58    return 0;
59  }
```

These man pages are included for your convenience.

NAME

strstr - locate a substring

SYNOPSIS

```
#include <string.h>

char *strstr(const char *haystack, const char *needle);
```

DESCRIPTION

The strstr() function finds the first occurrence of the substring needle in the string haystack. The terminating '\0' characters are not compared.

RETURN VALUE

The strstr() function returns a pointer to the beginning of the substring, or NULL if the substring is not found.

NAME

exit - cause normal program termination

SYNOPSIS

```
#include <stdlib.h>

void exit(int status);
```

DESCRIPTION

The exit() function causes normal program termination and the the value of status & 0377 is returned to the parent (see wait(2)). All functions registered with atexit() and on_exit() are called in the reverse order of their registration, and all open streams are flushed and closed. Files created by tmpfile() are removed.

The C standard specifies two defines EXIT_SUCCESS and EXIT_FAILURE that may be passed to exit() to indicate successful or unsuccessful termination, respectively.

RETURN VALUE

The exit() function does not return.

NAME

realloc - reallocate dynamic memory

SYNOPSIS

```
#include <stdlib.h>

void *realloc(void *ptr, size_t size);
```

DESCRIPTION

realloc() changes the size of the memory block pointed to by ptr to size bytes. The contents will be unchanged to the minimum of the old and new sizes; newly allocated memory will be uninitialized. If ptr is NULL, the call is equivalent to malloc(size); if size is equal to zero, the call is equivalent to free(ptr). Unless ptr is NULL, it must have been returned by an earlier call to malloc(), calloc() or realloc().

RETURN VALUE

realloc() returns a pointer to the newly allocated memory, which is suitably aligned for any kind of variable and may be different from ptr, or NULL if the request fails. If size was equal to 0, either NULL or a pointer suitable to be passed to free() is returned. If realloc() fails the original block is left untouched - it is not freed or moved.