Princeton University
COS 217: Introduction to Programming Systems
Fall 2006 Midterm Exam Answers

The exam was a 50 minute open-book open-notes exam.

Question 1 Part 1

A dangling pointer is a pointer pointing to a non-existent data object (e.g., freed memory in the heap segment, a local variable in an invalidated stack region, etc.). Leaked memory is dynamically allocated memory whose reference is lost and cannot be accessed.

Question 1 Part 2

An ADT is a specification of a set of data and the set of operations that can be performed on the data. Such a data type is abstract in the sense that it is independent of various concrete implementations. In C, an opaque pointer is a common choice to realize an ADT.

Question 1 Part 3

Any two of the following reasons:

- To provide each process with a large (2^{32} byte), contiguous memory.
- To provide each process with an image of exclusively protected memory.
- To obviate the need for each program to implement its own memory and storage handling routines.
- To enhance portability.

Question 1 Part 4

```
char *pc;
pc = (char*)malloc(strlen("Heap")+1);
assert(pc != NULL);
strcpy(pc, "Heap");
```

Question 1 Part 5

```
void f(void) {
   char pc[] = "Stack";
}
```

Question 1 Part 6

0121 is an octal constant whose decimal value is $1 * 8^2 + 2 * 8^1 + 1 * 8^0 = 64 + 16 + 1 = 81$. One can also convert it to a binary value, 01010001_2. Its decimal value can be also computed as $1 * 2^6 + 1 * 2^4 + 1 * 2^0 = 64 + 16 + 1 = 81$.

Question 1 Part 7

0xF0 is a hexadecimal constant whose binary representation is 11110000_2. Since the sign bit is 1, this is a negative integer. Therefore, by applying the "invert-and-plus-one" rule, one can obtain $-(00001111_2 + 1_2) = -(00010000_2) = -16$.

Question 1 Part 8

Any property that is synonymous with one of the following:

- Uniform (even, random) distribution over the hashing range
- Similar inputs shouldn’t generate similar hash values

Why? To minimize collisions.

Question 2

There are many valid approaches to this problem. Here are two simple ones.

Method 1: Algebraic

The value of the two's complement number $a_2a_1a_0 = -2^2a_2 + 2^1a_1 + 2^0a_0 = -4a_2 + 2a_1 + a_0$. Likewise, the value of the two’s complement number $b_3b_2b_1b_0 = -2^3b_3 + 2^2b_2 + 2^1b_1 + 2^0b_0 = -8b_3 + 4b_2 + 2b_1 + b_0$.

The single bit sign extension of the 3-bit two’s complement number $a_2a_1a_0$ is the two's complement number $a_2a_2a_1a_0$. Therefore, $b_3 = a_2$, $b_2 = a_2$, $b_1 = a_1$, and $b_0 = a_0$.

By substitution:

$b_3b_2b_1b_0 = a_2a_2a_1a_0 = -8a_2 + 4a_2 + 2a_1 + a_0 = -4a_2 + 2a_1 + a_0$

$a_2a_1a_0$ also equals $-4a_2 + 2a_1 + a_0$ (see above)

Therefore, the two's complement numbers $b_3b_2b_1b_0$ and $a_2a_1a_0$ represent the same value when $b_3b_2b_1b_0$ is the sign extension of $a_2a_1a_0$.

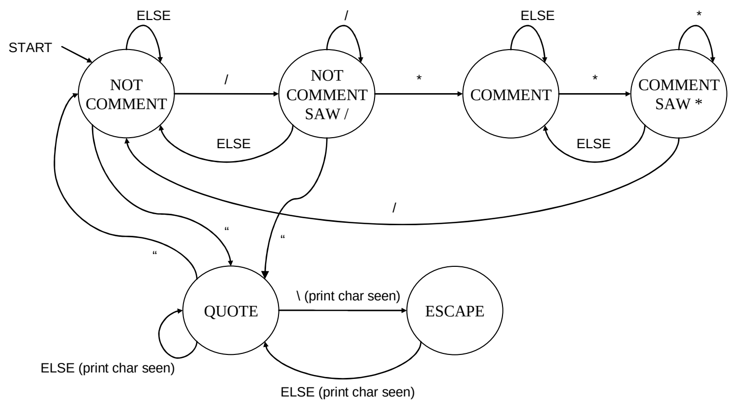
Method 2: Exhaustive

Let's enumerate all possible values of $b_3b_2b_1b_0$, the sign extension of $a_2a_1a_0$.

a value	**$a_2a_1a_0$**	**$b_3b_2b_1b_0$**	**b value**
3	011	0011	3
2	010	0010	2
1	001	0001	1
0	000	0000	0
-1	111	1111	-1
-2	110	1110	-2
-3	101	1101	-3
-4	100	1100	-4

Since the a value is the same as the b value for all settings of $a_2a_1a_0$, the two’s complement numbers $b_3b_2b_1b_0$ and $a_2a_1a_0$ represent the same value when $b_3b_2b_1b_0$ is the sign extension of $a_2a_1a_0$.

Question 3



Question 4 Part 1

The compiler generates the warning for line 37. This line declares fn as an array of 5 pointers to functions taking 3 parameters (of types char *, int, and char*, respectively) and returning an int. Additionally, this line initializes the first 4 elements of this array with the addresses of functions exit() (from the standard C library), insert_text(), search(), and print(). The warning comes from the assignment of the address of function exit() to the first element of fn. The reason is that exit() has void return type, and takes one formal parameter of type int.

Question 4 Part 2

Function insert_text() receives the buf pointer from main() and realloc()'s it. This may change the buffer's location in memory but, although the local pointer buf in insert_text() is updated properly, this value is not propagated back to main(). Therefore main()'s buf pointer will still point to the old location, which is no longer allocated to this buffer.

Question 4 Part 3

Here is partial list of correct answers:

(a) The code does not check the return value of malloc() (line 42) and realloc() (line 8) to make sure they are not NULL, i.e. to check that memory was properly allocated.
(b) The code does not check if the value of variable command provided by the user (line 48) is between 0 and 3. This can cause a segmentation fault at line 56.
(c) The code does not check for buffer overrun when reading the text variable (line 53). Reading a string with more that 99 characters can cause a segmentation fault.
(d) The call to realloc() (line 8) does not allocate enough space to store the old content of the string pointed by buf and the inserted text. This can cause a segmentation fault in lines 12 and 13.
(e) The code in function insert_text() does not make sure that the resulting string has the '\0' at the end. This can lead to segmentation fault when executing lines 24, 25 or 33.
(f) The call to printf() in line 33 is unsafe and can result in a segmentation fault if buf contains formatting sequences.
(g) Function print() fails to return a value, so a position out of the allocated buffer may be stored in pos (line 56), which can cause a segmentation fault later when using pos to index the buffer.

Question 4 Part 4

Same as the above.

Question 4 Part 5

The paragraph in lines 50-54 makes printing and exiting needlessly inconvenient for the user because it requires the user to input a string which is not necessary (and not used) for these options.

Question 4 Part 6

Here is a partial list of correct answers, besides the bugs listed in the items above:

(a) Lack of comments in the code.
(b) Lack of clarity in various parts of the code.
(c) Use of magic numbers in various places.
(d) Failure to include string.h and stdio.h
(e) Array fn is declared with more elements than necessary.
(f) Variable ch should be declared as int because it is assigned the result of getchar().
(g) In function insert_text(), variables used to index potentially large string should be of type size_t instead of int.
(h) Line 33 is unsafe; it should be printf("%s", buf);.
(i) There are unused arguments in functions search() and print().
(j) Function print() is declared to return an int but no value is returned.

Question 4 Part 7

Same as the above.