Princeton University
COS 217: Introduction to Programming Systems
Fall 2009 Midterm Exam Answers

The exam was a 50-minute, closed-book, closed-notes exam.

Question 1 Part A

```
Binary:  0110 1011
Hexadecimal:  6B
Two's complement:  1001 0101

Part A was worth 4 points.
```

Question 1 Part B

```
Suppose the original program runs for time unit 1.  After improvement, the execution time
should become 0.5. Assume we need to improve the time-consuming 20% of the code by factor
x:

0.8(1-x) + 0.2 = 0.5
=> x = 5/8

Part B was worth 4 points.
```

Question 1 Part C

Statement tests need to execute every possible statement. To statement test this code we need 2 sets of input data. One set tests one set of statements (for ex. 1, 3, 5), while the other test the other statements (for ex. 2,4,6). So the two sets have opposite values for the conditional statements.

Path tests execute all possible execution paths through the code. In this case it means that we execute every logically possible combination of statements. For this we need to test every possible combination of conditional values. There are 2 possible values for each conditional and 3 conditionals so there are 2^3 combinations. Therefore we need 8 sets of input data.

Part C was worth 4 points.

Question 1 Part D

```
No, the conditional test 0 <  j < 1 does not test whether j is between 0 and 1 because
that conditional is actually simplified to (0 < j) < 1 because of the rules of the <
operator (left-associative). This is what happens:
if (0 < j) is true then the expression becomes (0 < j) < 1 => 1 < 1 => false (0).
if (0 < j) is false then the expression becomes (0 < j) < 1 => 0 < 1 =>  true (1).

So, 0 < j < 1 actually tests whether j is negative or 0.

The correct test is:  (0 < j) && (j < 1)

Part D was worth 3 points.
```

Question 2 Part A

```
Each call of Stack_push() pushes "line" (alias "&line[0]") into the Stack object.  So
after breaking from the first loop, the Stack object contains three pointers, each of
which points to the "line" array.  The "line" array contains the final line read from
stdin, which is "Line three."  So, each item in the Stack object points to the string
"Line three."
```

Question 2 Part B

```
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"
enum {MAX_LINE_LENGTH = 256};
int main(void)
{
   char line[MAX_LINE_LENGTH];
   Stack_T s;
   char *p;
   s = Stack_new();
   while (fgets(line, MAX_LINE_LENGTH, stdin) != NULL) {
      p = (char*)malloc(strlen(line) + 1);
      if (p == NULL) {                          /* optional */
         fprintf(stderr, "Not enough memory.\n"); /* optional */
         return EXIT_FAILURE;                   /* optional */
      }                                         /* optional */
      strcpy(p, line);
      Stack_push(s, p);
   }
   while (! Stack_isEmpty(s)) {
      p = (char*)Stack_top(s);
      fputs(p, stdout);
      Stack_pop(s);
      free(p);
   }
   Stack_free(s);
   return 0;
}
```

Question 3

```
(1) 4
(2) 4
(3) 5
(4) 4
(5) 4
(6) 8
(7) 4
(8) 10
(9) 0
```

Note concerning part (6):

If the program is built on hats with gcc217, then the answer to part (6) is 8. The strlen() function traverses the 4 characters in ca2. But because ca2 is not null-terminated, strlen() continues traversing the memory that follows ca2. The ca1 array immediately follows ca2, so strlen() traverses the 4 additional characters in ca1, stopping only when it reaches the null character that terminates ca1. So strlen() traverses 8 characters in all, and thus returns 8.

In general, the answer to part (6) is system-dependent because (1) the C90 standard does not make any guarantees about the relative placement of local variables, and (2) traversing a string that is not null-terminated yields system-dependent results. On our system, strlen(ca2) returns 8. On other systems, strlen(ca2) could return some unsigned integer greater than or equal to 4. On yet other systems, strlen(ca2) could generate a segmentation fault. We accepted any reasonable answer to part (6).

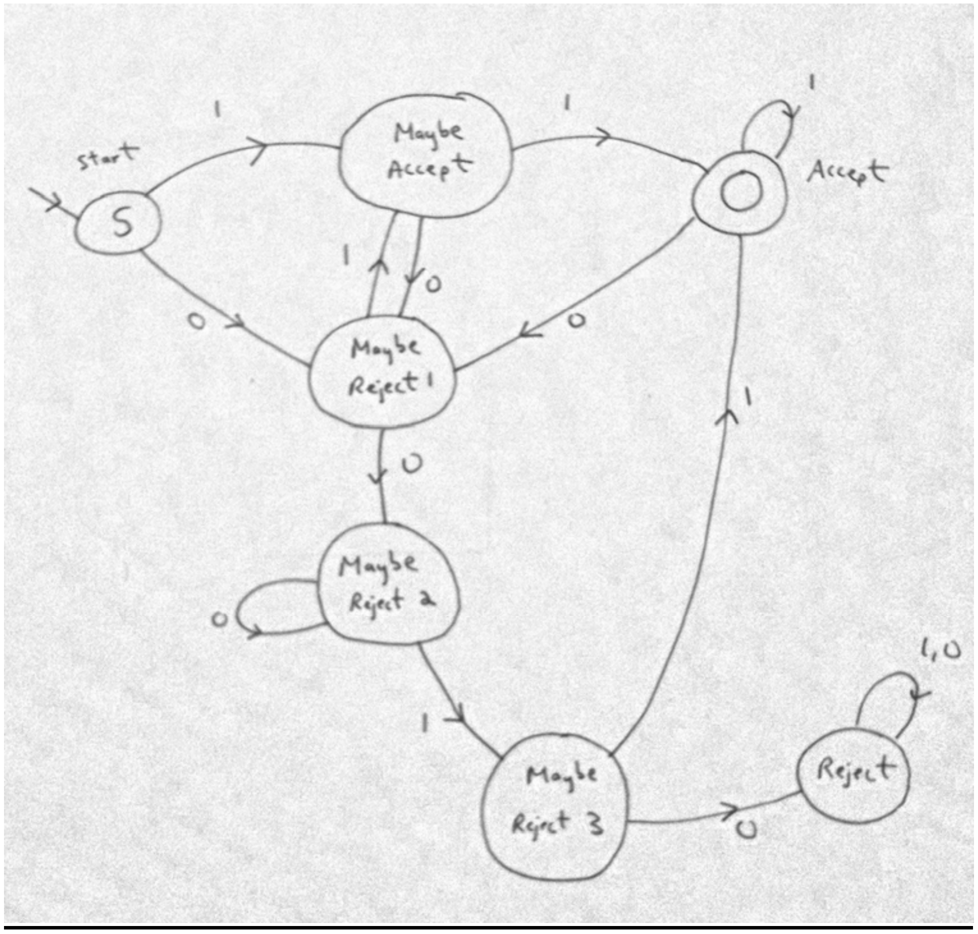
Note concerning part (9):

The answer to part (9) is 0 because ca3 is an uninitialized global variable, and so resides in the BSS section, and so is initialized to zeros. That is, each character of the array is set to '\0', the null character. So the strlen() function encounters the null character immediately, and returns 0.

Note concerning grading:

Parts (1)-(9) were worth 1 point each. Showing output with the proper format (that is, each answer preceded by an integer from 1 to 9 in parentheses) was worth 1 point.

Question 4



Question 5 Part A

```
Total:  6 points.

Yes, the code segment will compile (1 point); however, there will be compilation
warnings, specifically a type mismatch (1 point).  The code segment will execute and
print 3000 (the value of i) (1 point), the address of i (1 point), and the value at
memory address 3000 (1 point).  This last statement may result in a run-time error,
specifically a segmentation fault (1 point).
```

Question 5 Part B

```
i1=10
i2=20
*pi1=20
*pi2=10
```