Full Name: _____

Roll Number: _____

IT215: Systems Software, Winter 2014-15

First In-Sem Exam (2 hours)

February 5, 2015

Instructions:

- Make sure your exam is not missing any sheets, then write your name and roll number on the top of this page.
- Clearly write your answer in the space indicated. None of the questions need long answers.
- For rough work, do not use any additional sheets. Rough work will not be graded.
- Assume IA32 machine running Linux.
- The exam has a maximum score of 50 points. It is CLOSED BOOK. Notes are NOT allowed.
- Anyone who copies or allows someone to copy will receive F grade.

  Good luck!

1 (10):
2 (6):
3 (6):
4 (6):
5 (10):
6 (12):
TOTAL (50):

Problem 1. (10 points):

Circle the *single best* answer to each of the following questions. If you circle more than one answer, you will lose the mark for the corresponding question.

1. Which of the following has a larger size (in terms of bytes)?

   (a) pointer to a `char`

   (b) pointer to an `int`

   (c) pointer to a `float`

   (d) They are all of the same size.

2. Consider the following assembly instructions and C functions:

```
080483b4 <funcX>:
 80483b4: 55                    pushl  %ebp
 80483b5: 89 e5                 movl   %esp,%ebp
 80483b7: 8b 45 08              movl   0x8(%ebp),%eax
 80483ba: 8d 04 80              leal   (%eax,%eax,4), %eax
 80483bd: 8d 04 85 f6 ff ff ff  leal   0xfffffff6(,%eax,4), %eax
 80483c4: 89 ec                 movl   %ebp, %esp
 80483c6: 5d                    popl   %ebp
 80483c7: c3                    ret
```

   Circle the C function below that generates the above assembly instructions:

   (a)
```
int func1(int n) {
    return n * 20 - 10;
}
```

   (b)
```
int func2(int n) {
    return n * 24 + 6;
}
```

   (c)
```
int func3(int n) {
    return n * 16 - 4;
}
```

   (d) none of the above

3. Which of the following is NOT true of C switch statements compiled by `gcc`?

   (a) Sometimes a sequence of if-else statements is used to implement the switch statement.

   (b) Jump tables are used if there are multiple cases than span a small range of values.

   (c) Each entry in the jump table is guaranteed to be unique.

   (d) When a jump table is used, the switch execution time is independent of the number of cases.

For questions (4) and (5), consider the following code fragment.

```
unsigned short n = 0x1234;
unsigned char *p = (unsigned char *) &n;
printf("%d\n", *p);
```

4. What will the above code print on a little-endian machine such as a Pentium?

   (a) 12 (b) 18 (c) 34 (d) 52

5. What will the above code print on a big-endian machine?

   (a) 12 (b) 18 (c) 34 (d) 52

Problem 2. (6 points):

A C function `looper` and the assembly code it compiles to is shown below:

```
looper:
  pushl %ebp
  movl %esp,%ebp
  pushl %esi
  pushl %ebx
  movl 8(%ebp),%ebx
  movl 12(%ebp),%esi
  xorl %edx,%edx
  xorl %ecx,%ecx
  cmpl %ebx,%edx
  jge .L25
.L27:
  movl (%esi,%ecx,4),%eax
  cmpl %edx,%eax
  jle .L28
  movl %eax,%edx
.L28:
  incl %edx
  incl %ecx
  cmpl %ebx,%ecx
  jl .L27
.L25:
  movl %edx,%eax
  popl %ebx
  popl %esi
  movl %ebp,%esp
  popl %ebp
  ret
```

```
int looper(int n, int *a) {
  int i;
  int x = ________________;

  for(i = _____________;

      ____________________;

      i++) {

    if (_____________________)

       x = ____________________;


    ________________________;
  }

  return x;
}
```

Based on the assembly code, fill in the blanks in the C source code.

Notes:

- You may only use the C variable names `n`, `a`, `i` and `x`, not register names.
- Use array notation in showing accesses or updates to elements of `a`.

Problem 3. (6=3+3 points):

Consider the following C declaration:

```
struct s {
    char c;
    int i;
    char ca[3];
    double d;
} A[100];
```

A. How much space (in bytes) does the above array of structs take in memory? Draw (horizontally) the layout of the elements. Assume the Linux alignment rules discussed in class.

B. Can you modify the declaration to save space? If so, give the new declaration, and give the size of the modified array. Draw (horizontally) the new layout of the elements.

Problem 4. (6 points):

For problems A-C, draw the process tree and indicate how many "hello" output lines the program would print. (For space reasons, we are not checking error return codes, so assume that all functions return normally.) *Caution: Don't overlook the* `printf` *function in* `main`.

Problem A

```
void doit() {
  fork();
  fork();
  printf("hello\n");
  return;
}

int main() {
  doit();
  printf("hello\n");
  exit(0);
}
```

Answer: _____ output lines.

Problem B

```
void doit() {
  if (fork() == 0) {
    fork();
    printf("hello\n");
    exit(0);
  }
  return;
}

int main() {
  doit();
  printf("hello\n");
  exit(0);
}
```

Answer: _____ output lines.

Problem C

```
void doit() {
  if (fork() == 0) {
    fork();
    printf("hello\n");
    return;
  }
  return;
}

int main() {
  doit();
  printf("hello\n");
  exit(0);
}
```

Answer: _____ output lines.

Problem 5. (10 points):

Consider the C program below. (For space reasons, we are not checking error return codes, so assume that all functions return normally.)

```
int main () {
    if (fork() == 0) {
        if (fork() == 0) {
            printf("3");
        }
        else {
            pid_t pid; int status;
            if ((pid = wait(&status)) > 0) {
                printf("4");
            }
        }
    }
    else {
        printf("2");
        exit(0);
    }
    printf("0");
    return 0;
}
```

For each of the following strings, circle whether (Y) or not (N) this string is a possible output of the program. You will be graded on each sub-problem as follows:

- If you circle no answer, you get 0 points.
- If you circle the right answer, you get 2 points.
- If you circle the wrong answer, you get -1 points.

A. `32040` Y N

B. `34002` Y N

C. `30402` Y N

D. `23040` Y N

E. `40302` Y N

The next problem concerns the following C code. This program reads a string on standard input and prints an integer in hexadecimal format based on the input string it read.

```
#include <stdio.h>

/* Read a string from stdin into buf */
int evil_read_string()
{
    int buf[2];

    scanf("%s",buf);
    return buf[1];
}

int main()
{
    printf("0x%x\n", evil_read_string());
}
```

Here is the corresponding machine code on a Linux/x86 machine:

```
08048414 <evil_read_string>:
 8048414:  55                      push   %ebp
 8048415:  89 e5                   mov    %esp,%ebp
 8048417:  83 ec 14                sub    $0x14,%esp
 804841a:  53                      push   %ebx
 804841b:  83 c4 f8                add    $0xfffffff8,%esp
 804841e:  8d 5d f8                lea    0xfffffff8(%ebp),%ebx
 8048421:  53                      push   %ebx                    address arg for scanf
 8048422:  68 b8 84 04 08          push   $0x80484b8              format string for scanf
 8048427:  e8 e0 fe ff ff          call   804830c <_init+0x50>    call scanf
 804842c:  8b 43 04                mov    0x4(%ebx),%eax
 804842f:  8b 5d e8                mov    0xfffffff8(%ebp),%ebx
 8048432:  89 ec                   mov    %ebp,%esp
 8048434:  5d                      pop    %ebp
 8048435:  c3                      ret

08048438 <main>:
 8048438:  55                      push   %ebp
 8048439:  89 e5                   mov    %esp,%ebp
 804843b:  83 ec 08                sub    $0x8,%esp
 804843e:  83 c4 f8                add    $0xfffffff8,%esp
 8048441:  e8 ce ff ff ff          call   8048414 <evil_read_string>
 8048446:  50                      push   %eax                    integer arg for printf
 8048447:  68 bb 84 04 08          push   $0x80484bb              format string for printf
 804844c:  e8 eb fe ff ff          call   804833c <_init+0x80>    call printf
 8048451:  89 ec                   mov    %ebp,%esp
 8048453:  5d                      pop    %ebp
 8048454:  c3                      ret
```

Problem 6. (12 points):

This problem tests your understanding of the stack discipline and byte ordering. Here are some notes to help you work the problem:

- `scanf("%s", buf)` reads an input string from the standard input stream (stdin) and stores it at address `buf` (including the terminating '`\0`' character). It does **not** check the size of the destination buffer.
- `printf("0x%x", i)` prints the integer i in hexadecimal format preceded by "`0x`".
- Recall that Linux/x86 machines are Little Endian.
- You will need to know the hex values of the following characters:

Character	Hex value	Character	Hex value
'd'	0x64	'v'	0x76
'r'	0x72	'i'	0x69
'.'	0x2e	'l'	0x6c
'e'	0x65	'\0'	0x00
		's'	0x73

A. Suppose we run this program on a Linux/x86 machine, and give it the string "`dr.evil`" as input on stdin.

Here is a template for the stack, showing the locations of `buf[0]` and `buf[1]`. Fill in the value of `buf[1]` (in hexadecimal) and indicate where `ebp` points just **after** `scanf` returns to `evil_read_string`.

```
                |<- buf[0]->|<-buf[1] ->|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

What is the 4-byte integer (in hex) printed by the `printf` inside main?

`0x`_____

B. Suppose now we give it the input "`dr.evil.lives`" (again on a Linux/x86 machine).

(a) List the contents of the following memory locations just **after** `scanf` returns to `evil_read_string`. Each answer should be an unsigned 4-byte integer expressed as 8 hex digits.

`buf[0] = 0x`_____

`buf[3] = 0x`_____

(b) Immediately **before** the `ret` instruction at address `0x08048435` executes, what is the value of the frame pointer register `%ebp`?

`%ebp  = 0x`_____

You can use the following template of the stack as *scratch space*. *Note:* this does **not** have to be filled out to receive full credit.

```
                <- buf[0] -><- buf[1] ->
--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--
  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--
```