Full Name:_____

Univ ID:_____

Quiz I, Fall 2018 Date: Oct 9th, 2018

Instructions:

- This exam takes 60 minutes. Read through all the problems and complete the easy ones first.
- This exam is CLOSED BOOK. You may use a double-sided A4 cheatsheet that you've prepared yourself. Any electronic devices or other paper materials are forbidden.

1 (12)	2 (12)	3 (14)	4 (18)	5 (15)	6 (18)	7 (11)	Bonus (10)	Total (100+10)

Notice:

Unless otherwise noted, answer all questions in the quiz assuming a little-endian 64-bit x86 machine.

1 Bits, Bytes, Ints: (12 points)

Given a binary sequence `11111110`, answer the following questions (3 points each):

1. What is its most significant bit?

2. What is its decimal value when interpreting it as an `unsigned char`?

3. What is its decimal value when interpreting it as a (signed) `char`?

4. Please represent the sequence in the hexadecimal notation.

2 Byte ordering: (12 points)

Given the C code snippet below, answer the following questions (3 points each):

```
1: int c = 0x80000000;
2: char* cp;
3: cp = (char *)&c;
```

1. What is `c`'s size in bytes?

2. What is `c`'s value in decimal (Your answer may contain powers of 2, but not series.)?

3. What is the type of the expression `(*cp)`?

4. What is the value of the expression `(*cp)` in hexadecimal notation after executing line 3?

3 Floating point: (14 points)

Given the C code snippet below, answer the following questions (3 points each):

```
1: float f = -1.5;
2: char* cp;
3: cp = (char *)&f;
```

1. (3 points) What is `f`'s size in bytes?

2. (3 points) What is `cp`'s size in bytes?

3. (8 points) What are the values of `cp[0]`, `cp[1]`, `cp[2]`, `cp[3]` in hex notation after executing line 3?
   Value of `cp[0]`:

   Value of `cp[1]`:

   Value of `cp[2]`:

   Value of `cp[3]`:

4 Bitwise operation: (18 points)

Given the C program below, answer the following questions (3 point each):

```
// set the most significant bit of the integer pointed to by p to zero.
1: void clear_msb(unsigned int *p) {
2:    unsigned int mask = 0xffffffff;
3:    mask = mask >> 1;
4:    unsigned int x = _________;
5:    ________ = x;
6: }
7: int main() {
8:    float f = -1.5;
9:    int x = -1;
10:   clear_msb((int *)&f);
11:   clear_msb((unsigned int *)&x);
12:}
```

1. What is the value of `mask` (in hex notation) after executing line 3?

2. Complete line 4 to initialize `x` to have the same bit pattern as the integer pointed to by `p`, except that `x`'s most significant bit is always set to be zero. The righthand-side at line 3 must be an expression containing variable `mask`.

3. Complete line 5 to set the integer pointed to by `p` to have the same value as `x`.

4. What is the (decimal) value of the floating point variable `f` after executing line 10?

5. What is the value of int variable `x` (in decimal) after executing line 11? (Your answer may contain powers of 2, but not series.)

6. If one changes line 2 to `int mask = 0xffffffff;` what is the value of `mask` (in hex notation) after executing line 3?

5 More Pointers (15 points):

Given the C code snippet below, answer the following questions (3 point each):

```
1:  char c[8] = {0, 1, 2, 3, 4, 5, 6, 7};
2:  char *cp;
3:  cp = c;
4:  int *ip;
5:  ip = (int *)c;
```

1. Suppose the value of `c` is `0xff000000`, what is the value of `cp + 1` after executing line 3?

2. What is the value of expression `*(cp + 1)` after executing line 3?

3. What is an alternative (and equivalent) expression for `*(cp + 1)`?

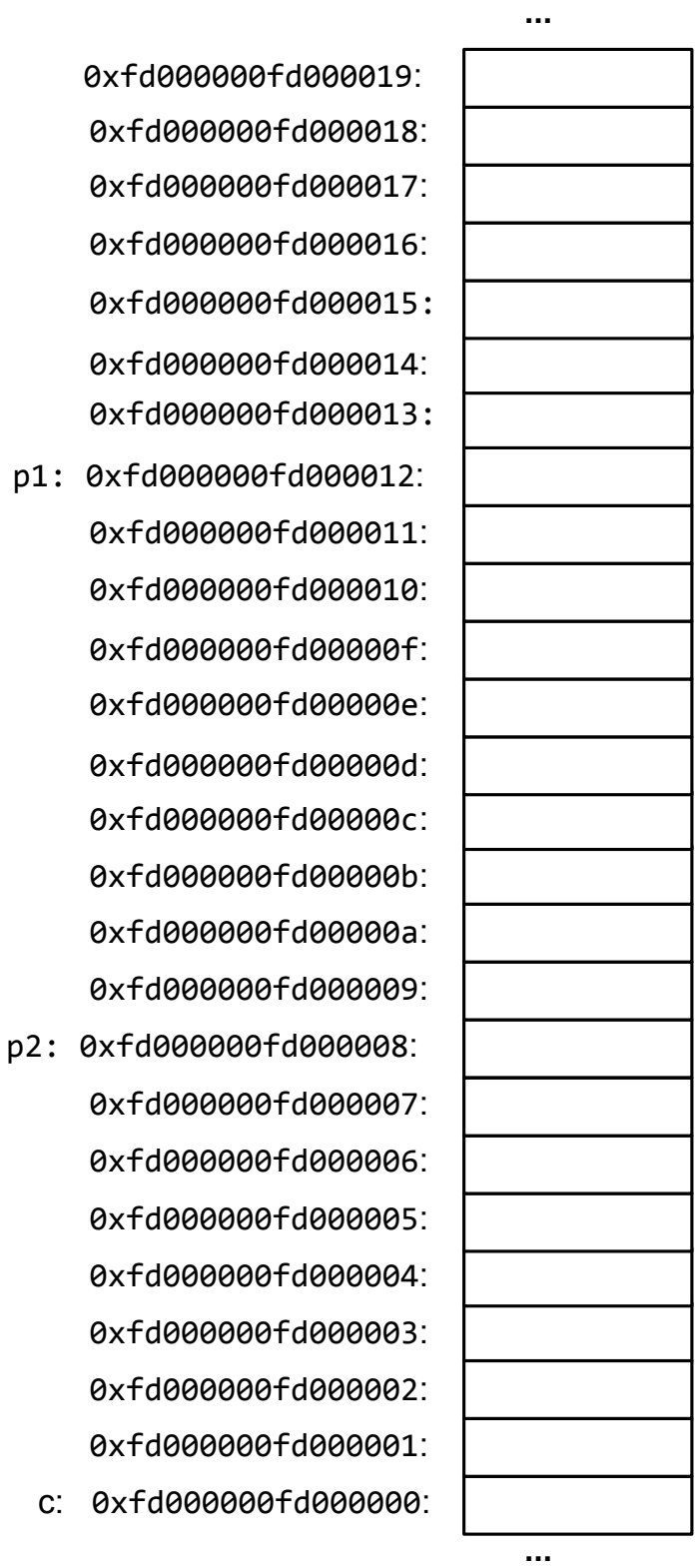
4. Suppose the value of `c` is `0xff000000`, what is the value of `ip + 1` after executing line 5?

5. What is the value of expression `*(ip + 1)` in hex notation after executing line 5?

6 String (18 points):

Consider the C program below.

```
1: int strlen(char *s) {
2:     int i = 0;
3:     while (s[i]) {
4:        i++;
5:     }
6:     return i;
7: }
8: void main() {
9:     char *p1;
10:    p1 = NULL;
11:    char *p2;
12:    char c = '\0';
13:    p2 = &c;
14:
15:    int l2 = strlen(p2);
16:    int l1 = strlen(p1);
17:}
```



The Figure on the right shows the memory being used by the program. Each hex number corresponds to the address of the memory cell next to it. The Figure shows where variables `p1`, `p2`, `c` are located in the memory. In particular, the address of each variable is shown next to the variable name. Note that the address of a variable is the address corresponding to the first byte of that variable.

1. (3 points) Mark the region of memory cells belonging to `p1`, `p2`, `c` in the figure.
2. (9 points) Write down the byte values contained in each memory cell belonging to variables `p1`, `p2` and `c`? Please use the hex notation.
3. (3 points) What is the result of executing line 15?

4. (3 points) What is the result of executing line 16?

7 Linked list (11 points):

In part5 of Lab1, you are asked to implement the following `list_find` function.

```
struct list_node {
   int value;
   struct list_node *next;
};

// Return a pointer to the first node in the given linked list
// (starting at head) with the specified value, and store the pointer
// to its predecessor node at predp.  If no such node exists,
// return NULL and set the predecessor to NULL as well.
struct list_node *
list_find(struct list_node *head, int value, struct list_node **predp)
{
   struct list_node *pred;
   struct list_node *node;

   pred = NULL;
   for (node = head; node != NULL; node = node->next) {









      pred = node;

   }



   predp = &pred;

   return node;
}
```

1. (5 points) Please complete the body of the `for` loop so that `node` points to the first node where the specified value is found or NULL when the loop exits.
2. (6 points) There is a bug in the implementation of `list_find`, please fix it.

8 Bonus: Programming (10 points):

Please complete the following code to implement a function `hex2int` that converts a hex string to its integer value (by interpreting the bit pattern represented by the hex string as 2's complement).

```
// return the integer value of the ASCII hex digit c
// you may assume hex digits are always given in lower case
char hex2digit(char c) {









}

int hex2int(char *s) {
    assert(strlen(s) == 8); //assume the string always contains exactly 8 hex "digits"









}

void main() {
  char *s;
  s = "ffffffff";
  int x = hex2int(s);
  assert(x == -1);
  s = "0000000f";
  x = hex2int(s);
  assert(x == 15);
}
```

1. (5 points) Please complete the helper function `hex2digit` that converts a hex character to an integer value in the range $[0, 15]$. You may assume that hex characters are always in lower case.
2. (5 points) Complete `hex2int` to convert a hex string representing an integer in 2's complement to the corresponding integer. The expected return value of this function is demonstrated using two example inputs in `main`. You may assume that the hex string is always exactly 8 characters long and in lower case.