Q1) [10 Points] **Negate** the following **nibble binary/hex** numbers, or write N/A if not possible. Remember to write your answer in the appropriate base. (A nibble is 4 bits)

<table>
<thead>
<tr>
<th>(Unsigned)</th>
<th>(Bias = -7)</th>
<th>(Bias = -7)</th>
<th>(Two's Comp)</th>
<th>(Two's Comp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b0101</td>
<td>0b0100</td>
<td>0xF</td>
<td>0b1100</td>
<td>0x10</td>
</tr>
<tr>
<td>0b N/A</td>
<td>0b1010</td>
<td>0x N/A</td>
<td>0b0100</td>
<td>0x6</td>
</tr>
</tbody>
</table>

...scratch space below...

Q2) [6 Points] Which of the following sums will yield an arithmetically incorrect result when computed with two’s complement nibbles?

<table>
<thead>
<tr>
<th>Correct ☐</th>
<th>Incorrect ☐</th>
<th>Correct ☐</th>
<th>Incorrect ☐</th>
<th>Correct ☐</th>
<th>Incorrect ☐</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xD + 0xE + 0xF</td>
<td>0x7 + 0x8</td>
<td>0x3 + 0x5</td>
<td>0x3 + 0x5</td>
<td>0x3 + 0x5</td>
<td>0x3 + 0x5</td>
</tr>
</tbody>
</table>

...scratch space below...

Q3) [12 Points] For each of the following representations, what is the **fewest number of bits** needed to cover the given range, which is inclusive of the endpoints (e.g., [1, 4] is the numbers 1, 2, 3 and 4). Write “N/A” if it is impossible. For the **Bias Value** (final value = unsigned + bias value), we’ll let YOU specify whatever offset you wish to minimize the total number of bits needed for the Bias encoding.

<table>
<thead>
<tr>
<th>Range</th>
<th>Unsigned</th>
<th>One’s Comp</th>
<th>Two’s Comp</th>
<th>Sign&amp;Mag</th>
<th>Bias</th>
<th>Bias Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 0, 10 ]</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>[ -4, -1 ]</td>
<td>N/A</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>[ 1, 4 ]</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

...scratch space below...
For this page, assume all mallocs are successful, all necessary libraries are #included, and any heap accesses outside what the program allocates is a segmentation fault.

Q4) [12 Points] Which of the following are possible, if perhaps unlikely, results of attempting to compile and run this code? (select ALL that apply)

```c
int main() {
    int32_t *str = (int32_t *) malloc(sizeof(int32_t) * 3);
    printf("%s", (char *) str); // A char is 8 bits.
    return 0;
}
```

- Compilation error due to invalid typecast
- Runtime typecasting error
  - A segmentation fault
  - The program prints the empty string
  - The program prints CS61C
- The program prints CS61C rocks!

Q5) [10 Points] Each of the following evaluate to an address in memory. In other words, they "point" somewhere. Where in memory do they point?

<table>
<thead>
<tr>
<th>Code</th>
<th>Static</th>
<th>Stack</th>
<th>Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>arr[0]</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>dest</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>dest[0]</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>&amp;arrPtr</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

What are the first 8 characters the program prints? (Note: The program DOES compile and run without error.)

This function makes all letters capital.

```
// The ASCII values for 'A', 'B', etc. are 65, 66, ... Important
// The ASCII values for 'a', 'b', etc. are 97, 98, ... Important
char *arr[] = {"Go", "Bears"};
```

```
int main() {
    char **arrPtr = arr;
    char *dest[2];
    int j;
    for (int i = 0; i < 2; i++) {
        char *currString = *arrPtr;
        dest[i] = (char *) malloc(strlen(currString) + 1);
        for (j = 0; j < strlen(currString); j++) {
            dest[i][j] = currString[j] & ~(1 << 5); // ⇐ Hint: Focus on this line!≠
        }
        dest[i][j] = '\0';
        arrPtr++;
    }
    printf("%s %s", dest[0], dest[1]);
}
```