Memory Management in C

1) In which memory sections (code, static, heap, stack) do the following reside?

<table>
<thead>
<tr>
<th>arg</th>
<th>arr</th>
<th>str</th>
<th>val</th>
</tr>
</thead>
</table>

2) What are two reasons we might need to use malloc in a C program?

3) What is wrong with the C code below:

```c
int* ptr = malloc(4 * sizeof(int));
if(extra_large) {
    ptr = malloc(10 * sizeof(int));
}
return ptr;
```

4) For the singly linked list implementation below, fill out free_ll, which frees all of the memory allocated for the linked list.

```c
typedef struct {
    ll_node* next;
    int element;
} ll_node;

void free_ll(ll_node* list) {
}
```

5) Write append_ll which appends a new node with element x. Assume list is not null. Return success.

```c
int append_ll(ll_node* list, int x) {
```

6) You have two ll_node pointers. Write a function that swaps the two pointers.
**MIPS cheat sheet (including instructions you might not have seen yet)**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>add dest, src0, src1</td>
<td>add $s0, $s1, $s2</td>
</tr>
<tr>
<td>sub</td>
<td>sub dest, src0, src1</td>
<td>sub $s0, $s1, $s2</td>
</tr>
<tr>
<td>addi</td>
<td>addi dest, src0, immediate</td>
<td>addi $s0, $s1, 12</td>
</tr>
<tr>
<td>sll / srl</td>
<td>sll dest, src, immediate</td>
<td>sll $t0, 4($s0)</td>
</tr>
<tr>
<td>lw / lb</td>
<td>lw dest, offset(base addr)</td>
<td>lw $t0, 4($s0)</td>
</tr>
<tr>
<td>sw / sb</td>
<td>sw src, offset(base addr)</td>
<td>sw $t0, 4($s0)</td>
</tr>
<tr>
<td>bne</td>
<td>bne src0, src1, branchAddr</td>
<td>bne $t0, $t1, notEq</td>
</tr>
<tr>
<td>beq</td>
<td>beq src0, src1, branchAddr</td>
<td>bne $t0, $t1, Eq</td>
</tr>
<tr>
<td>j</td>
<td>j jumpAddr</td>
<td>j jumpWhenDone</td>
</tr>
<tr>
<td>jr</td>
<td>jr reg</td>
<td>jr $ra</td>
</tr>
</tbody>
</table>

### C

```c
// $s0 -> a (use $s0 for a),
// $s1 -> b
// $s2 -> c, $s3 -> z

int a=4, b=5, c=6, z;
z = a+b+c+10;

// $s0 -> int *p = (int *)malloc
// (3*sizeof(int));
// $s1 -> a
p[0] = 0;
int a = 2;
p[1] = a;
p[a] = a;

// $s0 -> a, $s1 -> b
int a = 5, b = 10;
if (a + a == b) {
    a = 0;
} else {
    b = a - 1;
}
```

### MIPS

```mips
addi $s0, $0, 0
addi $s1, $0, 1
addi $t0, $0, 30
loop: beq $s0, $t0, done
add $s1, $s1, $s1
addi $s0, $s0, 1
j loop
done: # done!
```