Memory Access Pattern

1. For the Barcelona (64KB, 2-way set-associative cache with 64-byte blocks), estimate the miss rate of the two following code fragments: (assume ints are 4-bytes data types and variables $i$ and $j$ are register allocated.)

```c
int i, j, a[100];
for (i = 0 ; i < 100000 ; i ++) {
    for (j = 0 ; j < 100 ; j ++) {
        a[j] ++;
    }
}
```

```c
int i, j, a[100000];
for (i = 0 ; i < 100 ; i ++) {
    for (j = 0 ; j < 100000 ; j ++) {
        a[j] ++;
    }
}
```
2. For a 16kB direct-mapped, write-back cache with 32B blocks on a machine with 32-bit address spaces (both virtual and physical), consider the following code:

```c
struct thing {
    char *name;
    int size;
    int color;
    int weight;
};

struct thing things[100];
// fill in things...
enum colors {RED, BLUE, GREEN, YELLOW, NUM_COLORS};
// loop 1
int how_heavy_are_things_this_color[NUM_COLORS] = {0};
for (int color = 0 ; color < NUM_COLORS ; color++) {
    for (int i = 0 ; i < 100 ; i++) {
        if (things[i].color == color) {
            how_heavy_are_things_this_color[color] += things[i].weight;
        }
    }
}
// loop 2
struct thing *first_thing_called_foobar = NULL;
for (int i = 0 ; i < 100 ; i++) {
    if (strcmp("foobar", things[i].name) == 0) {
        first_thing_called_foobar = &things[i];
        break;
    }
}
```

Assume that integers and pointers are 4 bytes big. Assume that color and i are register allocated. Assume that the things array starts at a cache block boundary.

(a) Describe the trace of addresses generated by loop 1.
(b) Assuming no hardware prefetching, compute how many cache misses would occur for loop 1?
(c) How would this differ if some form of hardware prefetching was implemented? Estimate the likely number of cache misses.
(d) Estimate the number of misses for loop 2 when a match is never found (i.e., when the loop does all 100 iterations). Assume no hardware prefetching and that the cache starts empty. Furthermore, assume that “foobar” maps to a different part of the cache than the array things or the name strings. Finally, assume that the name strings are completely contained within one cache block, but they are equally likely to map to any index of the cache.