3M decides to make Post-Its by printing yellow squares on white pieces of paper. As part of the printing process, they need to set the CMYK (cyan, magent, yellow, black) value for every point in the square. 3M hires you to determine the efficiency of the following algorithms on a machine with a 2048-byte direct-mapped data cache with 32-byte blocks. You are given the following definitions:

```c
struct point_color {
    int c;
    int m;
    int y;
    int k;
};
```

```c
struct point_color square[16][16];
```

Assume the following:

- `sizeof(int) == 4`.
- `square` begins at memory address 0.
- The cache is initially empty.
- The only memory accesses are to the entries in the array `square`. Variables `i` and `j` are stored in registers.

Determine the cache performance of the following code:

```c
for (i=0; i<16 ; i++){
    for (j=0; j<16 ; j++){
        square[i][j].c = 0;
        square[i][j].m = 0;
        square[i][j].y = 1;
        square[i][j].k = 0;
    }
}
```

1. What is the total number of writes?
2. What is the total number of writes that miss in the cache?
3. What is the miss rate?

(Problem 6.26 from Bryant/O’Hallaron’s textbook)
Given the assumptions in Problem 6.25, determine the cache performance of the following code:

```
for (i=0; i<16 ; i++){
    for (j=0; j<16 ; j++){
        square[j][i].c = 0;
        square[j][i].m = 0;
        square[j][i].y = 1;
        square[j][i].k = 0;
    }
}
```

1. What is the total number of writes?
2. What is the total number of writes that miss in the cache?
3. What is the miss rate?

(Problem 6.27 from Bryant/O’Hallaron’s textbook)
Given the assumptions in Problem 6.25, determine the cache performance of the following code:

```
for (i=0; i<16 ; i++){
    for (j=0; j<16 ; j++){
        square[j][i].y = 1;
    }
}
for (i=0; i<16 ; i++){
    for (j=0; j<16 ; j++){
        square[j][i].c = 0;
        square[j][i].m = 0;
        square[j][i].k = 0;
    }
}
```

1. What is the tonal number of writes?
2. What is the total number of writes that miss in the cache?
3. What is the miss rate?

*(Problem 6.28 from Bryant/O’Hallaron’s textbook)*

You are writing a new 3D game that you hope will earn you fame and fortune. You are currently working on a function to blank the screen buffer before drawing the next frame. The screen you are working with is a 640×480 array of pixels. The machine you are working on has a 64 KB direct-mapped cache with 4-byte lines. The C structures you are using are:

```c
struct pixel {
    char r;
    char g;
    char b;
    char a;
};

struct pixel buffer[480][640];
int i, j;
char *cptr;
int *iptr;
```

Assume the following:

- `sizeof(char) == 1` and `sizeof(int) == 4`.
- `buffer` begins at memory address 0.
- The cache is initially empty.
- The only memory accesses are to the entries in the array `buffer`. Variables `i`, `j`, `cptr`, and `iptr` are stored in registers.

What percentage of writes in the following code will miss the cache?

```c
for(j=0; j<640; j++){
    for(i=0; i<480; i++){
        buffer[i][j].r = 0;
        buffer[i][j].g = 0;
        buffer[i][j].b = 0;
        buffer[i][j].a = 0;
    }
}
```
(Problem 6.29 from Bryant/O’Hallaron’s textbook)
Given the assumptions in Problem 6.28, what percentages of writes in the following code will miss the cache?

```c
1 char *cptr = (char *) buffer;
2 for(; cptr < (((char *) buffer) + 640 * 480 * 4); cptr++)
3   *cptr = 0;
```

(Problem 6.30 from Bryant/O’Hallaron’s textbook)
Given the assumptions in Problem 6.28, what percentages of writes in the following code will miss the cache?

```c
1 char *iptr = (char *) buffer;
2 for(; iptr < ((int *) buffer + 640 * 480); cptr++)
3   *iptr = 0;
```