◆◆ Problem – week 16

This is another problem that tests your ability to analyze the cache behavior of C code. Assume we execute the three summation functions in Figure 1 under the following conditions:

- The machine has a 256 direct-mapped cache with a 16-byte block size.
- Within the two loops, the code uses memory accesses only for the array data.
- The loop indices and the value sum are held in registers.
- Array a is stored starting at memory address SRAM 0x0000 (m=10).

Fill in the table for the approximate **cache miss rate** y **AMAT** for the two cases $N = 10$ and $N = 8$.

	N=10	AMAT (ns)	N=8	AMAT (ns)
sumA				
sumB				
sumC				

```
typedef short array_t[N][N];

short sumA(array_t a)
{
    int i, j;
    int sum = 0;
    for (j = 0; j < N; j++)
        for (i = 0; i < N; i++) {
            sum += a[i][j];
        }
    return sum;
}
short sumB(array_t a)
{
    int i, j;
    short sum = 0;
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++) {
            sum += a[i][j];
        }
    return sum;
}

short sumC(array_t a)
{
    int i, j;
    int sum = 0;
    for (j = 0; j < N; j += 2)
        for (i = 0; i < N; i += 2) {
            sum += (a[i][j] + a[i + 1][j]
                + a[i][j + 1] + a[i + 1][j + 1]);
        }
    return sum;
}
```

Cache size	Block size	SRAM	SDRAM
0,5KB	8B	3ns	80ns
0,5KB	16B	5ns	85ns
1KB	8B	7ns	83ns
1KB	16B	11ns	89ns