◆◆ Problem - week 15 (17-05-2023)

Name 1 ______________________ **Name 2** ______________________

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 32-bytes cache (Direct Mapped) with a 8-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 0x00 (m=8).

Fill in the table for the approximate **cache miss rate** for the two cases *N* = 6 and *N* = 4.

	Miss Rate (N=4)	Miss Rate (N=6)
sumA		
sumB		

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

short sumA(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 sumB(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 array_t[4][4]				B	m/h	m/h
i	j		Address			
0	0	0	00000000			
0	1	2	00000010			
0	2	4	00000100			
0	3	6	00000110			
1	0	8	00001000			
1	1	10	00001010			
1	2	12	00001100			
1	3	14	00001110			
2	0	16	00010000			
2	1	18	00010010			
2	2	20	00010100			
2	3	22	00010110			
2	0	24	00011000			
2	1	26	00011010			
2	2	28	00011100			
2	3	30	00011110			
3	0	32	00100000			
3	1	34	00100010			
3	2	36	00100100			
3	3	38	00100110			

https://www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/start.html

short array_t[6][6]				B	m/h [i,j]	m/h [j,i]
i	j		Address			
0	0	0	00000000			
0	1	2	00000010			
0	2	4	00000100			
0	3	6	00000110			
0	4	8	00001000			
0	5	10	00001010			
1	0	12	00001100			
1	1	14	00001110			
1	2	16	00010000			
1	3	18	00010010			
1	4	20	00010100			
1	5	22	00010110			
2	0	24	00011000			
2	1	26	00011010			
2	2	28	00011100			
2	3	30	00011110			
2	4	32	00100000			
2	5	34	00100010			
3	0	36	00100100			
3	1	38	00100110			
3	2	40	00101000			
3	3	42	00101010			
3	4	44	00101100			
3	5	46	00101110			
4	0	48	00110000			
4	1	50	00110010			
4	2	52	00110100			
4	3	54	00110110			
4	4	56	00111000			
4	5	58	00111010			
5	0	60	00111100			
5	1	62	00111110			
5	2	64	01000000			
5	3	66	01000010			
5	4	68	01000100			
5	5	70	01000110			