1.
What is loop unrolling? How can it make your program more efficient? How can it make your program less efficient?

2.
What affects the data stored on the stack? In the context of the attack lab, what instructions should be paid careful attention to?

3.
The provided function `func_one` takes as input two pointers, that are actually each individually pointing to the first element in a N by M array of integers.

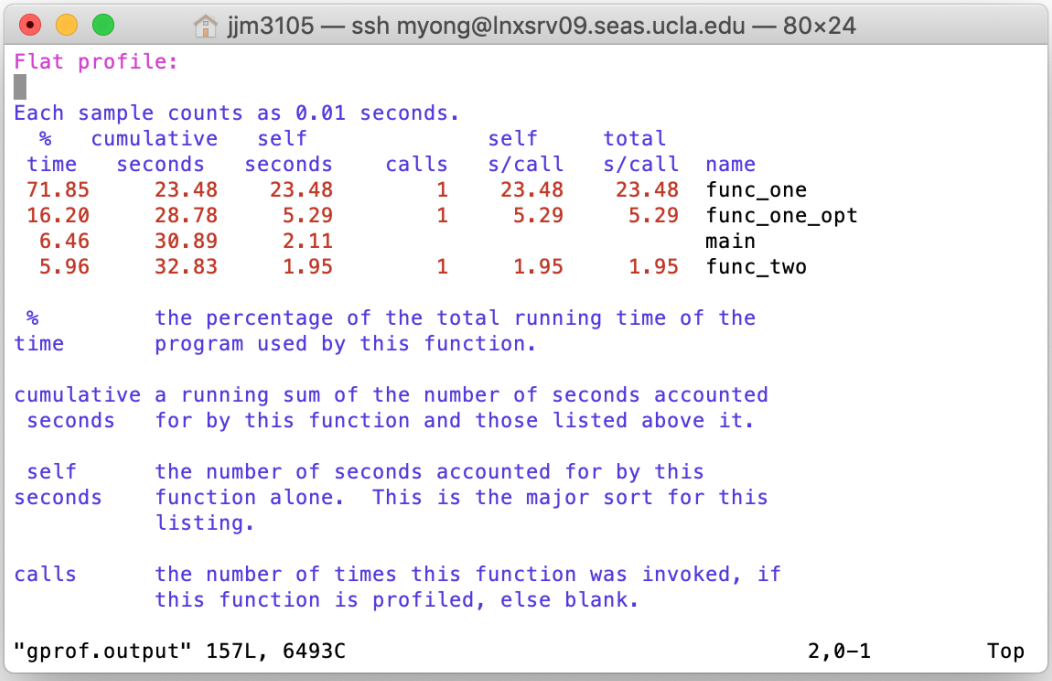
```
int func_one(char* one, char* two, int N, int M) {
                int i, j, k;
                int sum = 0;

                char* ptr1 = one;
                char* ptr2 = two;

                for (k = 0; k < 4; k++) {
                   for (j = 0; j < M; j++) {
                         for (i = 0; i < N; i++) {

                               char one = *(ptr1 + k + j*4 + i*4*M);
                               int masked = one & 0xFF;

                               int shift = k << 3;
                               int shifted = masked << shift;

                               *(ptr2 + k + j*4 + i*4*M) = masked;
                               sum += shifted;
                         }
                   }
                }

                return sum;
}
```

In what ways can we optimize the above function?

4.
The provided code below is an optimization of the previous problem. Fill in the blanks.

```
int func_two(char* one, char* two, int N, int M) {
                int i, j, k;
                int sum = 0;
                int temp = 0;

                char* ptr1 = one;
                char* ptr2 = two;

                for (i = 0; i < N; i++) {
                   for (j = 0; j < M; j++) {
                         temp = (0xFF & *ptr1);
                         sum += _______________;
                         *ptr2 = _______________;
                         ptr1++;
                         ptr2++;

                         temp = _______________;
                         sum += _______________;
                         *ptr2 = _______________;
                         ptr1++;
                         ptr2++;

                         ______________________;
                         ______________________;
                         ______________________;
                         ______________________;
                         ______________________;

                         ______________________;
                         ______________________;
                         ______________________;
                         ______________________;
                         ______________________;
                   }
                }

                return sum;
}
```

FYI.



The above shows the running time of the functions discussed in problems 5 and 6. `func_one` is the code from problem 5 as is, `func_one_opt` is one optimization of `func_one`, and `func_two` is the completed code from problem 6.

The results were generated using gprof