Name: __________________________________

**Theoretical Questions:**

1. Assume that the compiler always uses the stack to pass parameters and return results. Sketch how the stack would look like after calling function foo. Explain the most frequently used target for a buffer overflow and how it affects program execution. Explain how canaries can prevent this exploit.

```c
int process_Input( char * message, unsigned int length)
// We trust that length is the correct message length
{
    char buffer[1024];
    if( 1024 – length < 0 )   // Check for buffer overflow
    {
        strcpy(buffer, message);
        return -1;
    } else {
        for( ; buffer ; buffer++)
        {
            if(!isalpha(*buffer))
                *buffer="#";
        }
        return 1;
    }
```
2. The function in Problem 1 makes a very clumsy attempt at checking for buffer overflow. Why is the check on line 5 useless?

3. A software house decides to replace all calls to strcpy() and strcat() with strncpy() and strncat(). Explain why this can result in truncation and why this can create additional vulnerabilities.

4.