General Approaches to CS61 exams and life:
When faced with a long/difficult question use form hypothesizes/guesses and test through scientific examination.

Example: label the following data structures with their corresponding assembly code.

- array
- array of array pointers
- linked list
- binary tree

Solution: examine their complexity and number of assembly code lines. Examine the number of calls. It is NOT necessary/efficient to try and completely understand the assembly code for a question like this.

stacksmash:
This is a type of attack when someone tries to overwrite the return address of a function. This can happen if someone is able to write more data than can fit into a buffer. This can be achieved for example if the function gets is used.

![Diagram of stacksmash]

Attempting to stacksmash!
1) If we try a buffer overflow attack on a buffer that uses gets. Gets is dangerous because it will read an unlimited amount of data without returning an error. However, when we use gets to try and overwrite the return address and get an error meaning our attempt was foiled. How???

Answer: gets actually uses get_chk that ensures that the size is big enough to fit the amount of data we are trying to write.

2) We can then get around this by using a function like read_line that calls get_s. This way, gets is masked and not identified by the compiler. See below

```c
read_line(char* buffer) {
    if (gets(buffer))
        return 1;
    else
        return 0;
}
```

We still get an error and are thus foiled. How???

Answer: Gcc keeps us safe using thread local storage area. %gs stores the canary in between buffer and return address at some random place: this way, buffer overflow will modify the canary and the program will quit.