Call stack / run time stack

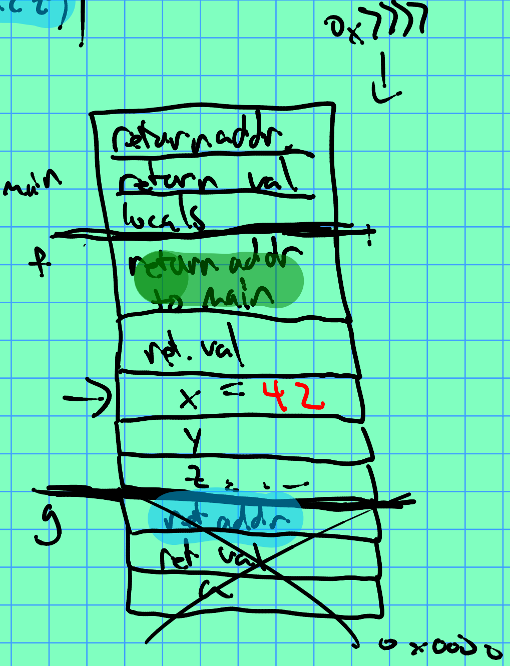
Each time a function is called, a new "frame" is created to store relevant info about the call.
The frame includes
- storage for local variables + parameters
- storage for return value
- return address.

(This is how function calls are managed behind the scenes.)

```
int f(int x) {
    int y,z;
    ;
z = g(x*y);
 cout << "g returned" << z;
 return 0;
}

main() {
  f(42);
  cout << "called f!";
```

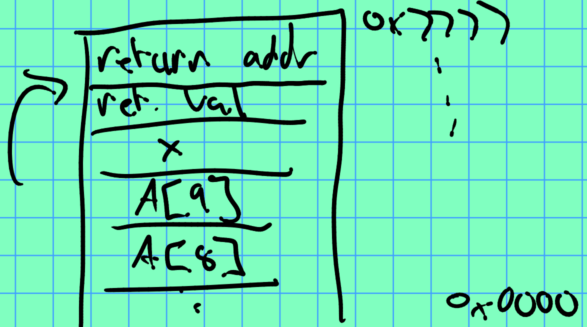
```
int g(int a) {
   return a*a;
}
```



upon g finishing, its frame will be removed.

Security implications!

```
int f(...) {
    int x;
    int A[10];
```



Question: what happens if we write "off the end" of A? I.e., write to A[10], A[11], ...