Dynamic Memory:

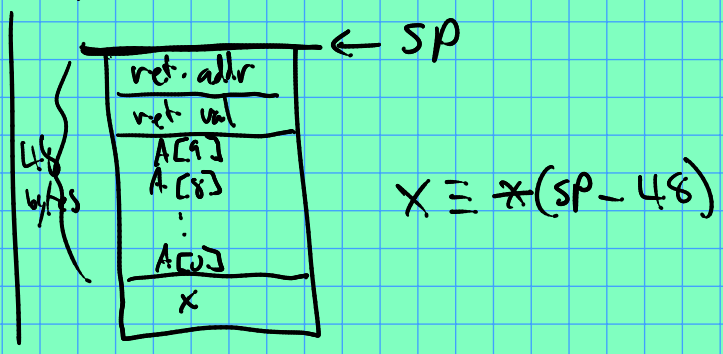
— Allocate memory as the program runs.

(e.g. vectors expand upon push_back)

— Why is this an issue?

Recall the stack layout:

```
void f(int n) {
  int A[n];
  int x;
  :
  :
```

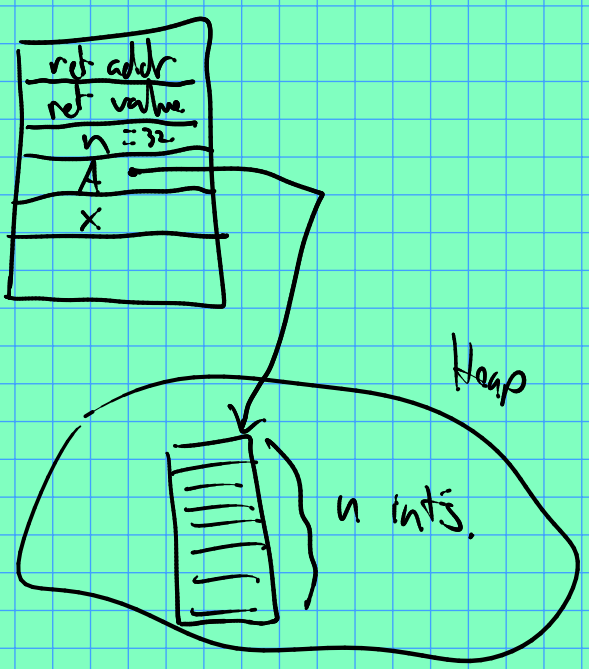


If size of A not known at compile time, no way to know the label/address/offset that defines x.

Instead, use dynamic memory allocations.

Better:

```
void f(int n) {
  int* A = new int[n];
  int x;
  :
  :
```



What does "new" do?

① Finds contiguous block of memory.

(this happens in libc, & maybe the OS via system calls in libc.)

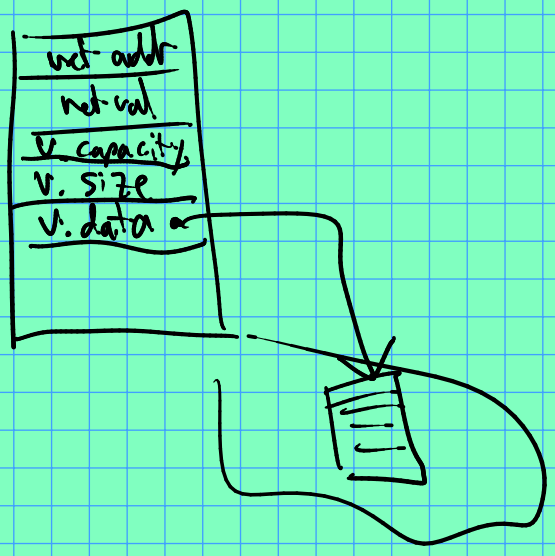
(2) Block of memory is marked as "reserved" for your program.

(3) You are given the address of the new block.

datatype of (new char) is char*, (new int) is int*, etc.

What about this:

```
f() {
  vector<int> V;
    :
```



Note: You should return memory when you're done with it!

Here's how:

```
int* A = new int[n];
   :
   // use A...
// Now free the memory:
delete A;
```

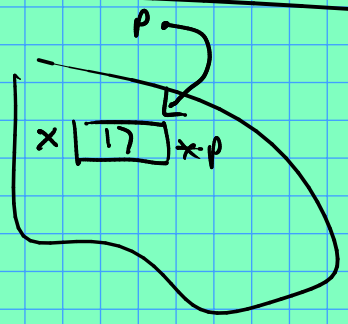
"unmarks" the block as used.

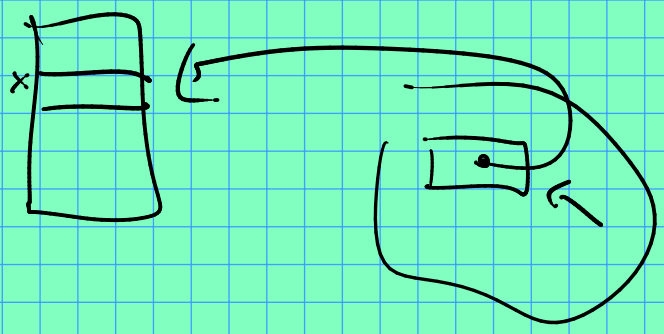
Note: — End of a function will **not** automatically delete dynamic allocations.

— End of program **will**.

Still — imagine a long-running process like a web server. If dynamic memory wasn't freed, could eventually consume all of main memory x_x.

```
int * p = new int;
*p = 17;
int& x = *p;
```






```
int x
int ** p = new int*;
*p = &x;
```