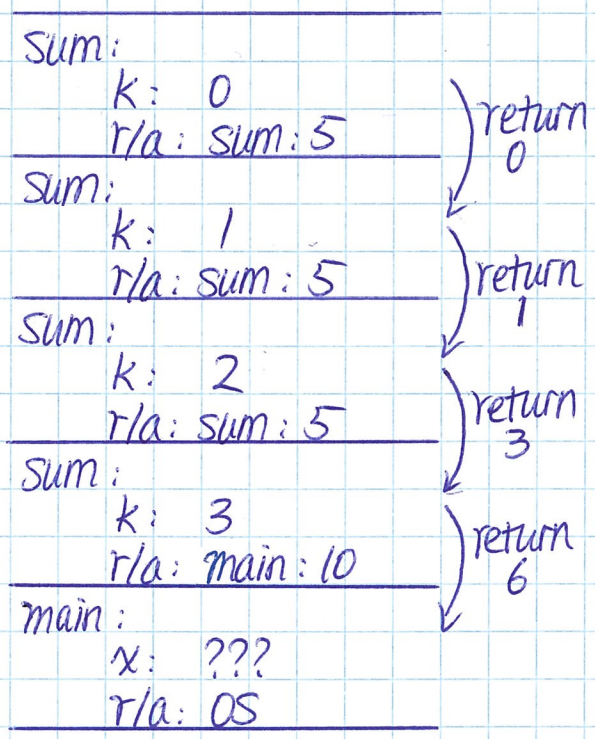
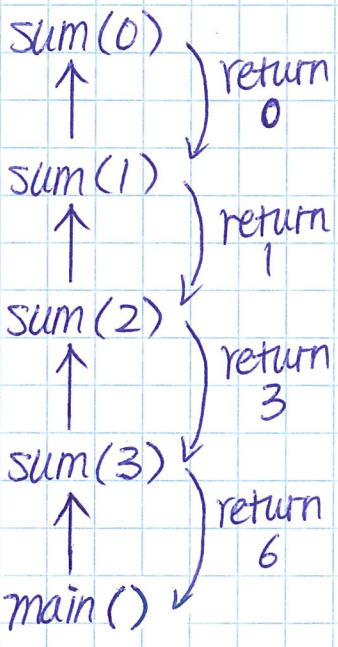
Section 8 Space complexity (Slides 55-59)

```
1  int sum (int k) {
2      if (k<=0) {
3          return 0;
4      }
5      return k + sum (k-1);
6  }
7
8  int main(void) {
9      int x = sum(3);
10 }
```





- Every stack frame contains a # to be added to the sum.
- We need every frame.

The sum function
- make a recursive call first
- use the return value of the recursive call to calculate the result.

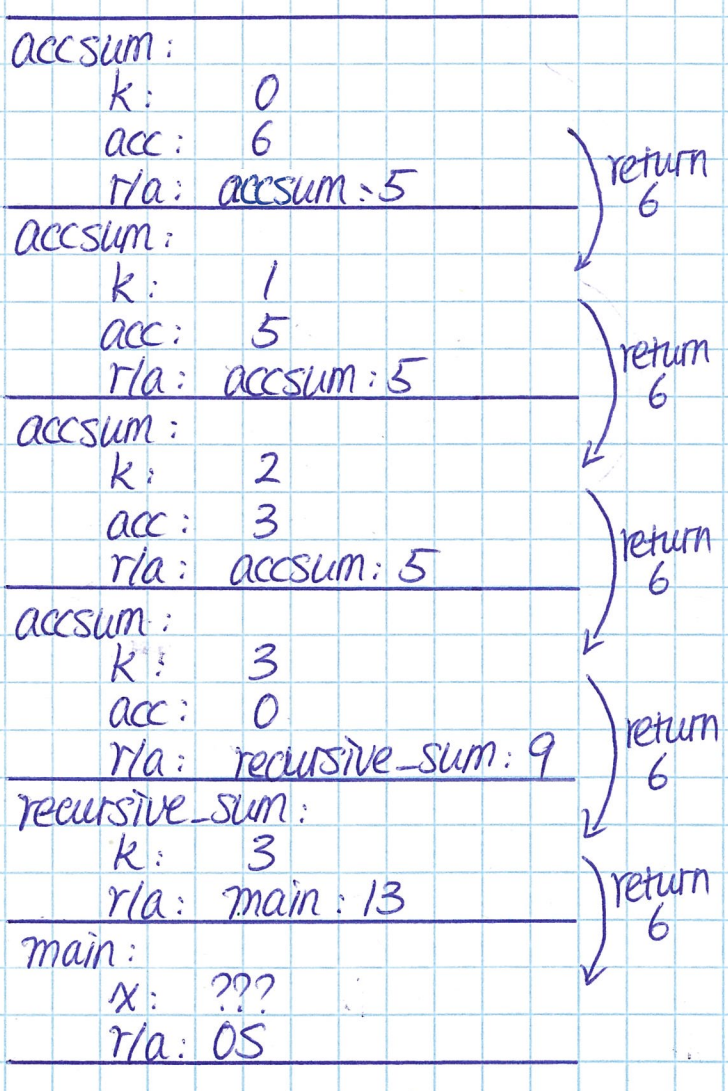
Section 8 Space complexity (Slides 55-59)

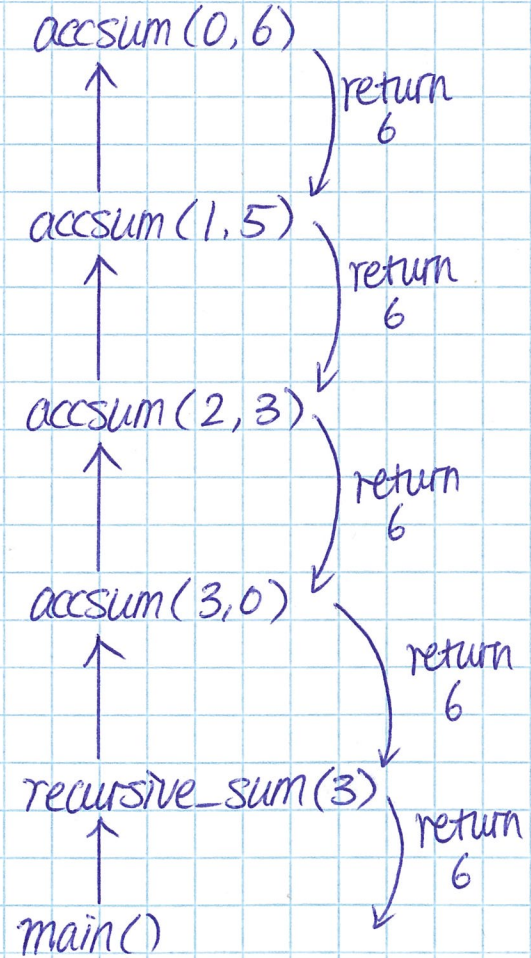
```
1   int accsum(int k, int acc) {
2       if (k == 0) {
3           return acc;
4       }
5       return accsum(k-1, k+acc);
6   }
7
8   int recursive_sum(int k) {
9       return accsum(k, 0);
10  }
11
12  int main(void) {
13      int x = recursive_sum(3);
14  }
```

The accsum function
- perform calculation first.
- execute the recursive call.
- pass the result of the current step to the next recursive step.

The return value of any recursive step is the same.

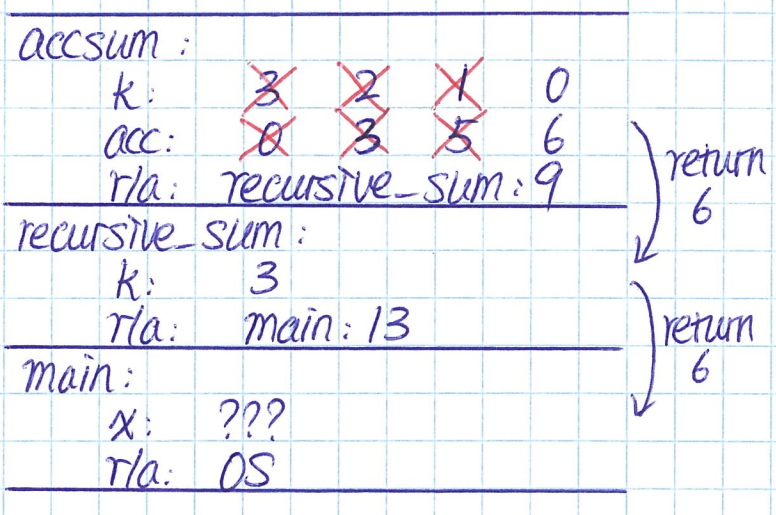
Consequence:
- don't need the current stack frame for the next recursive step.
- reuse the current stack frame for the next recursive step.

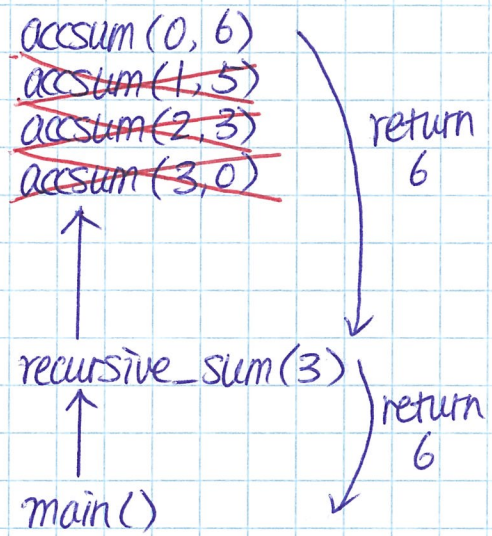




Section 8 Space complexity (Slides 55-59)

```
1   int accsum(int k, int acc){
2       if (k==0){
3           return acc;
4       }
5       return accsum(k-1, k+acc);
6   }
7
8   int recursive_sum(int k){
9       return accsum(k, 0);
10  }
11
12  int main(void){
13      int x = recursive_sum(3);
14  }
```





Tail recursion:
- can be transformed into a loop.
- could in principle run forever. (# of stack frames does not grow.)