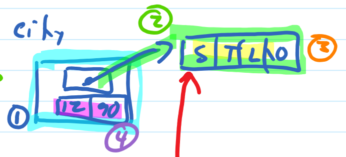
Resizing an array

① `city c;` allocates a city struct on the stack

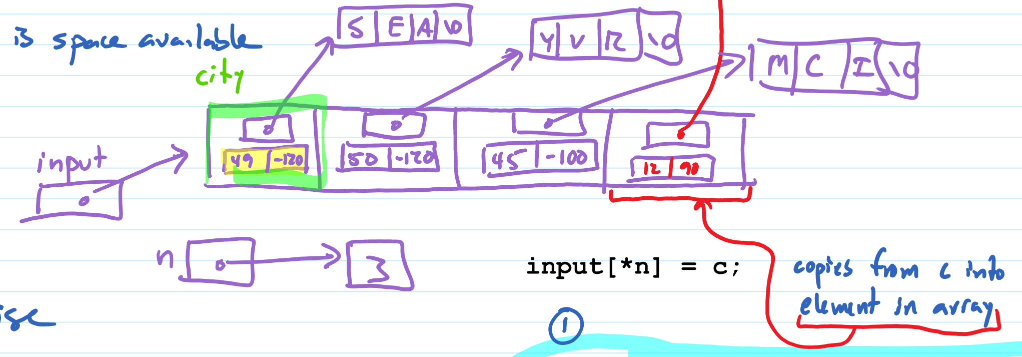
② `c.name = malloc(sizeof(char) * 4);`
points the name to an array on the heap

③ `strcpy(c.name, code);`
copies into that array

④ `find_city(c.name, &c.coord);`
finds coords and saves them in the struct



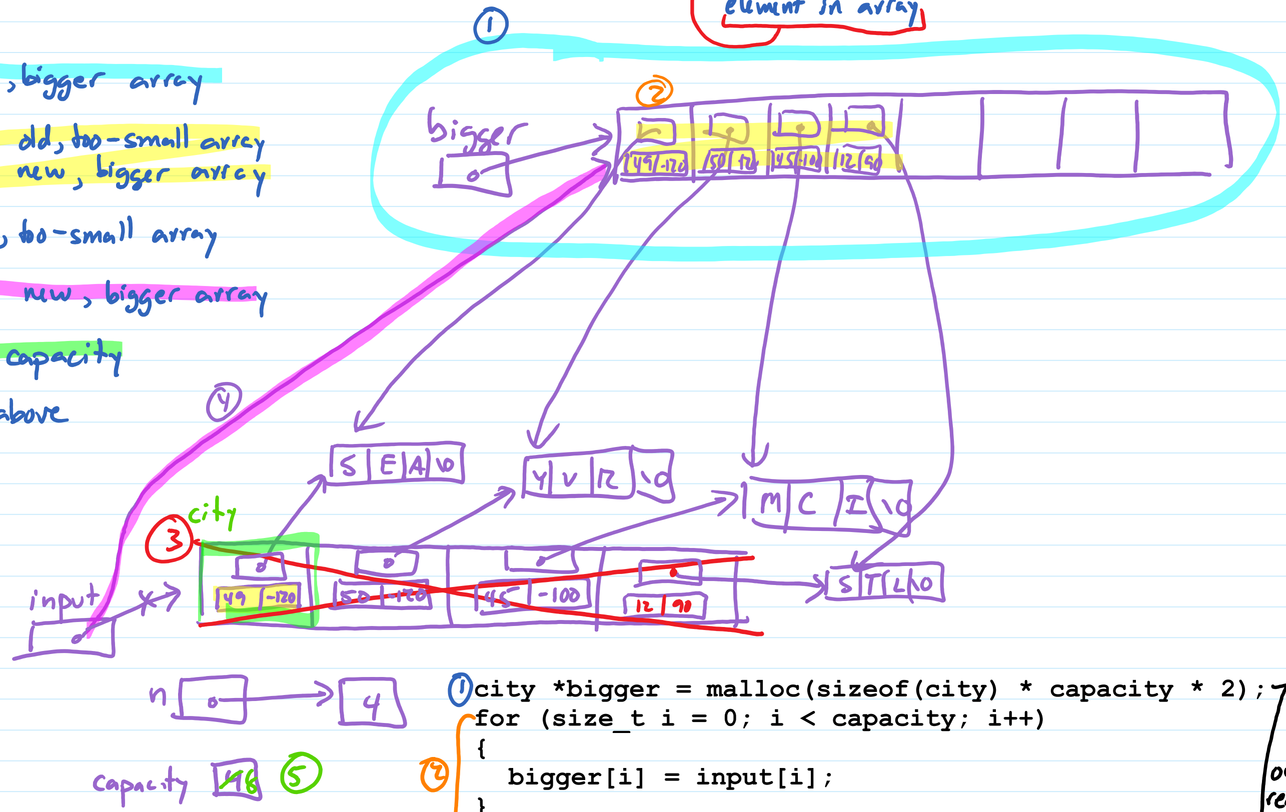
if there is space available



`input[*n] = c;` copies from c into element in array

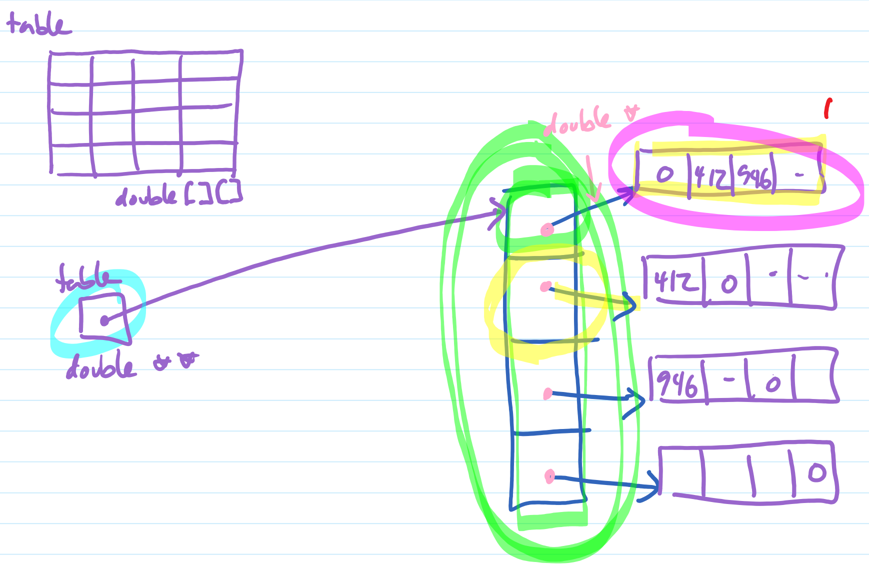
otherwise

1) make new, bigger array
2) copy from old, too-small array to new, bigger array
3) free old, too-small array
4) remember new, bigger array
5) update capacity
6) do the above



```
①city *bigger = malloc(sizeof(city) * capacity * 2);
  for (size_t i = 0; i < capacity; i++)
② {
    bigger[i] = input[i];
  }
③free(input);
④input = bigger;
⑤capacity *= 2;
```

or realloc



start w/ capacity 1 (size 0)

add 1-by-1 until size 16)

count copies for increase-by-1 vs increase-by-double

Abstract Data Type (ADT)

$\hookrightarrow$ spec of operations on some data structure   (<u>header file</u>)

<u>List of Cities ADT</u>

make empty list

find current size

add city to end of list

get city at index

destroy list