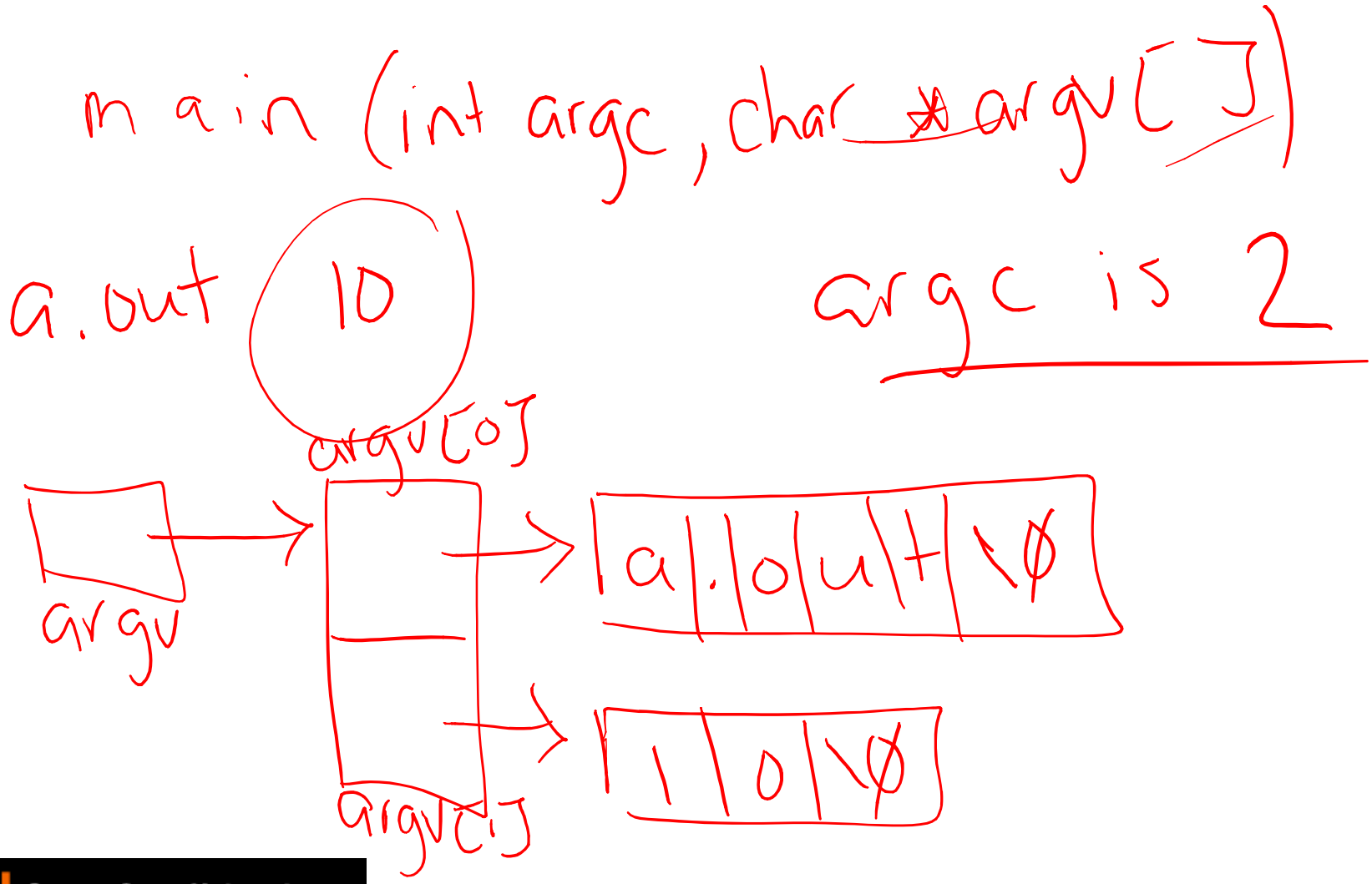
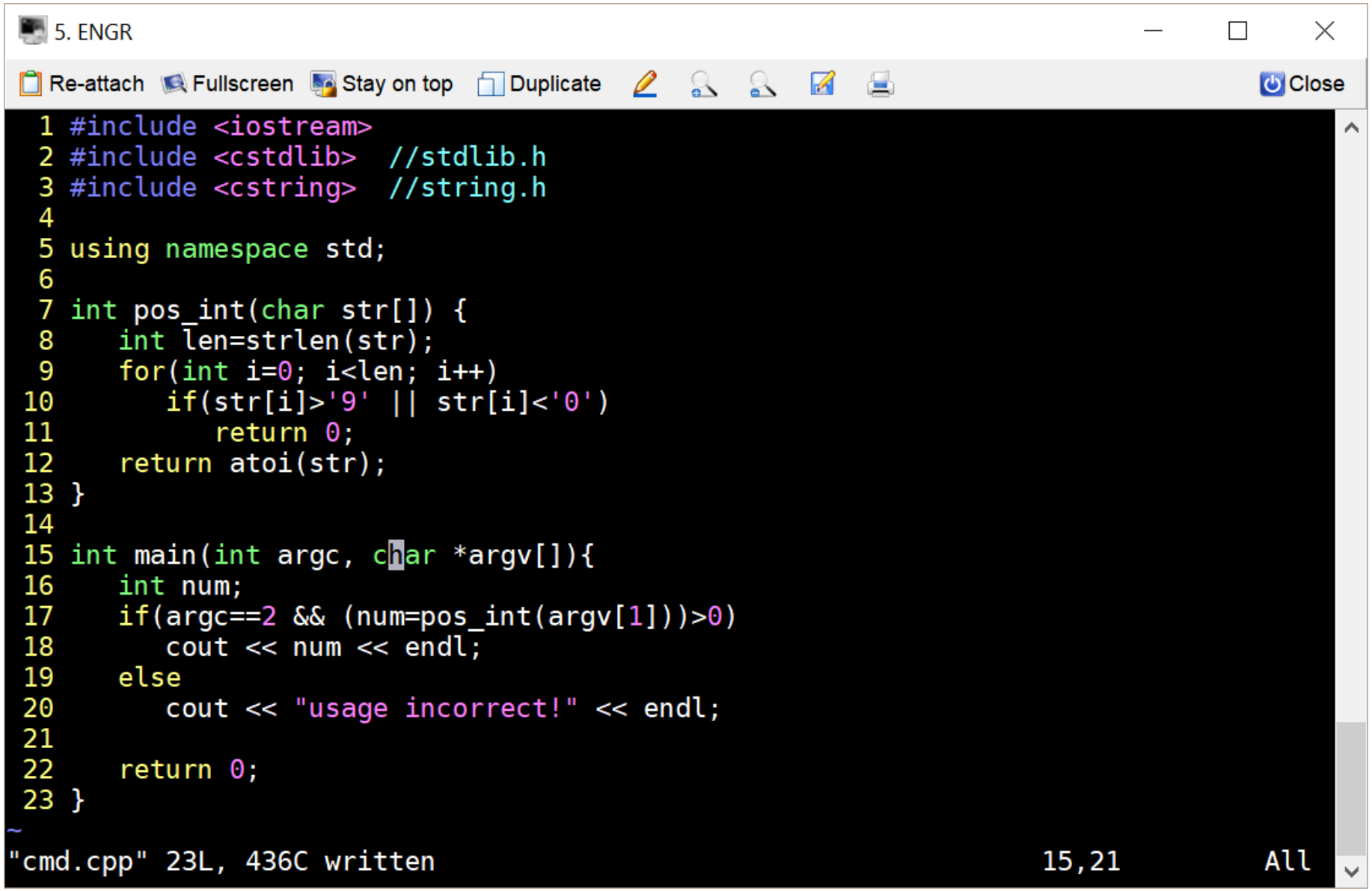
CS 161
Intro to CS I

2d arrays and Command-Line Arguments

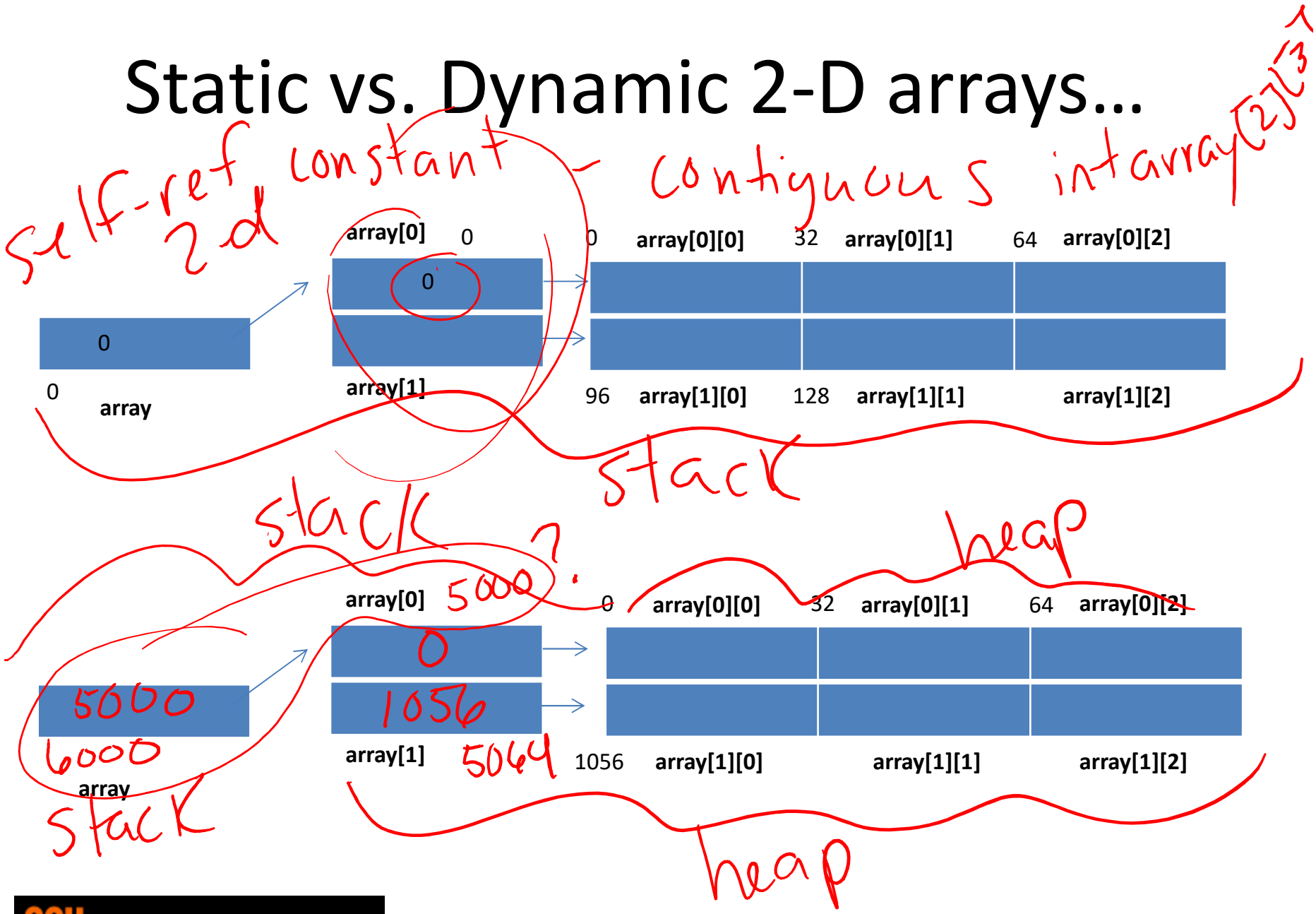
OSU Oregon State University

Finish Command Line Args…



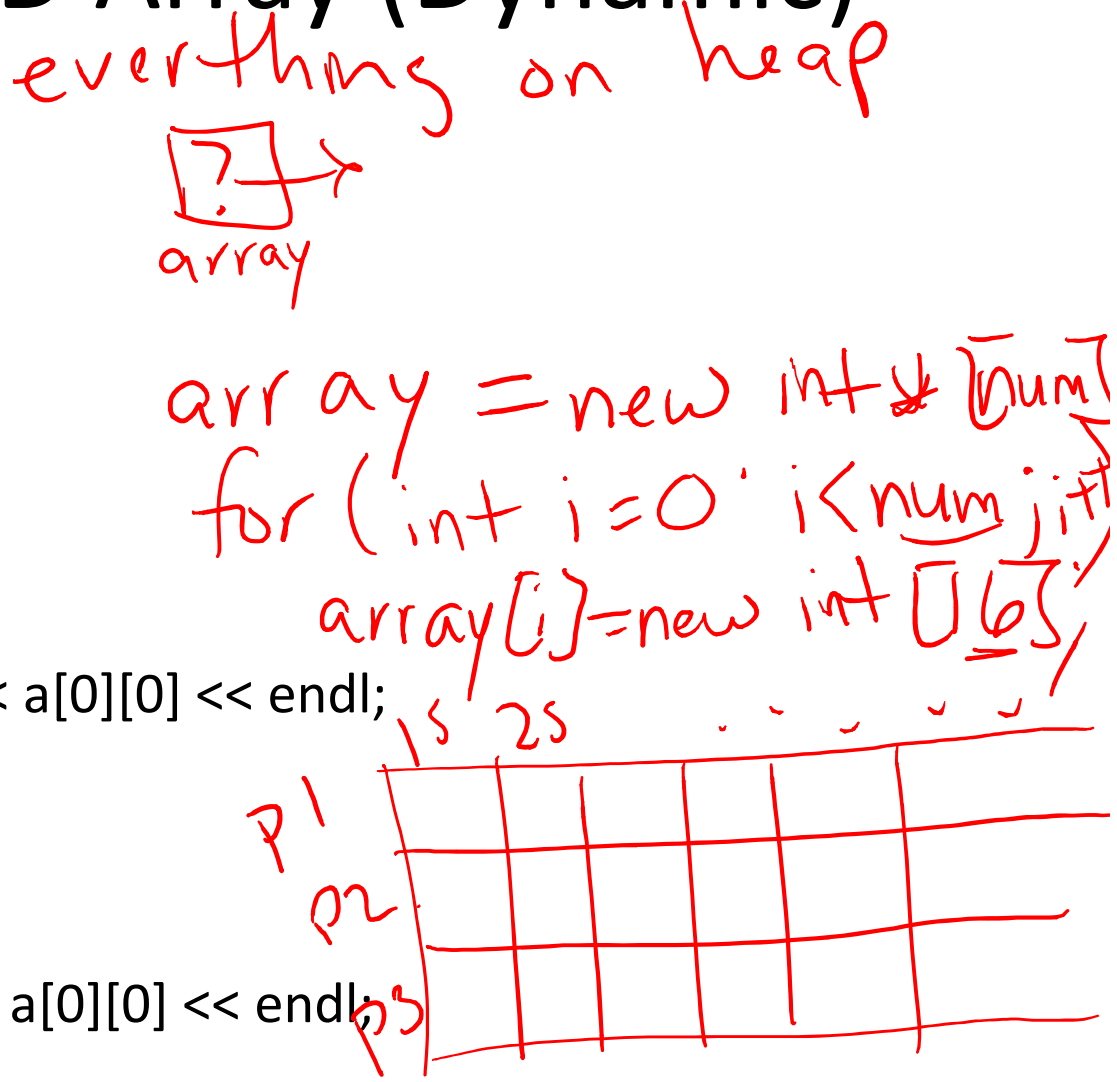

5. ENGR
Re-attach
Fullscreen
Stay on top
Duplicate
Close
1 #include <iostream>
2 #include <cstdlib> //stdlib.h
3 #include <cstring> //string.h
4
5 using namespace std;
6
7 int pos_int(char str[]) {
8   int len=strlen(str);
9   for(int i=0; i<len; i++)
10   if(str[i]>'9' || str[i]<'0')
11   return 0;
12   return atoi(str);
13 }
14
15 int main(int argc, char *argv[]){
16   int num;
17   if(argc==2 && (num=pos_int(argv[1]))>0)
18   cout << num << endl;
19   else
20   cout << "usage incorrect!" << endl;
21
22   return 0;
23 }
~
"cmd.cpp" 23L, 436C written
15,21
All

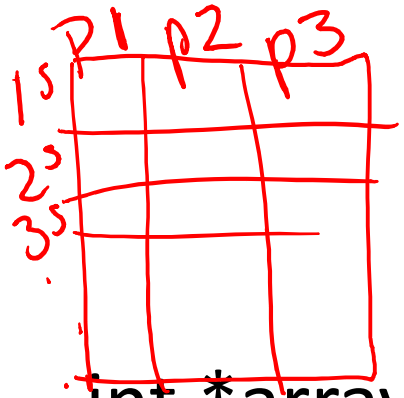
Static vs. Dynamic 2-D arrays…



Passing a 2-D Array (Dynamic)

```
int main() {
  int **array;
  …
  pass_2darray(array);
  …
}
void pass_2darray(int *a[]) {
   cout << “Array at zero: ” << a[0][0] << endl;
}
OR
void pass_2darray(int **a) {
  cout << “Array at zero: ” << a[0][0] << endl;
}
```

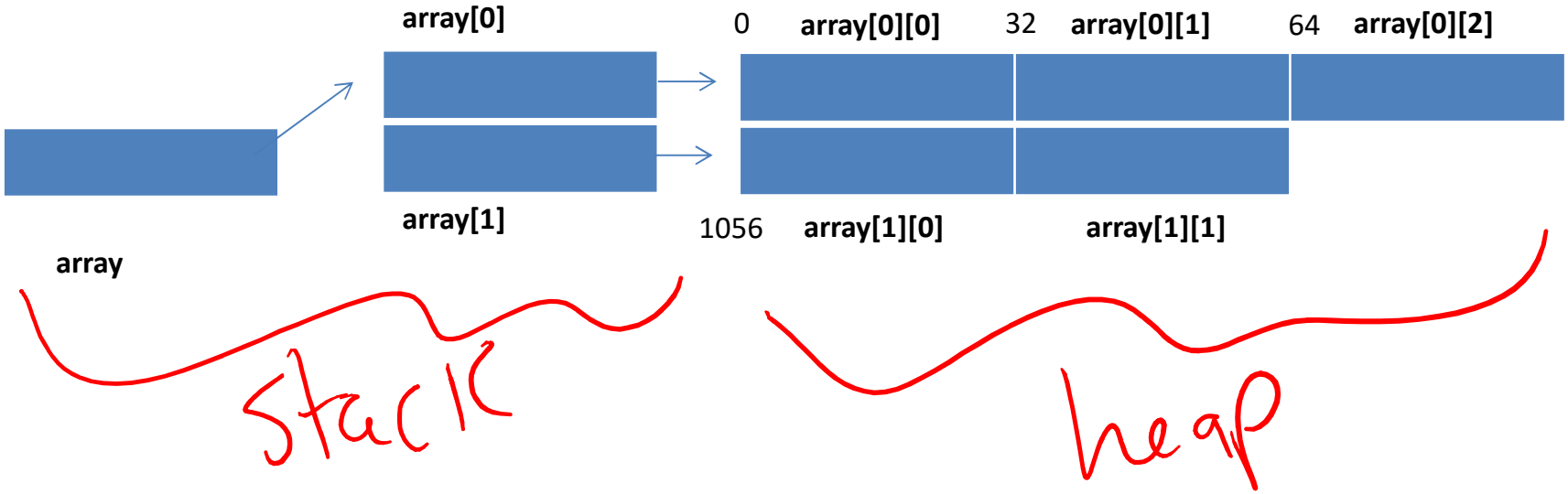




Jagged Arrays

```
int *array[2];
array[0] = new int[3];
array[1] = new int[2];
```

int * array[16]
for all rows/16 of them
array[i] = new int[cnt]



Passing a 2-D Array (Dynamic)

```
int main() {
   int *array[2];
   …
   pass_2darray(array);
   …
}
void pass_2darray(int *a[]) {
    cout << “Array at zero: ” << a[0][0] << endl;
}
```

OR

```
void pass_2darray(int **a) {
   cout << “Array at zero: ” << a[0][0] << endl;
}
```

Create 2-D Array in Functions

```
int main() {
   int **array;
   …
   array = create_2darray(rows, cols);
   …
}
int **create_2darray(int r, int c) {
    int **a;
    a = new int*[r];
    for(int i=0; i<r; i++)
        a[i] = new int[c];
    return a;

}
```

Create 2-D Array in Functions

```
int main() {
  int **array;
  …
create_2darray(&array, rows, cols);
  …
}
void create_2darray(int ***a, int r, int c) {
   *a = new int*[r];
   for(int i=0; i<r; i++)
      (*a)[i] = new int[c];
}
```

Create 2-D Array in Functions

```
int main() {
   int **array;
   …
create_2darray(array, rows, cols);
   …
}
void create_2darray(int **&a, int r, int c) {
    a = new int*[r];
    for(int i=0; i<r; i++)
        a[i] = new int[c];
}
```

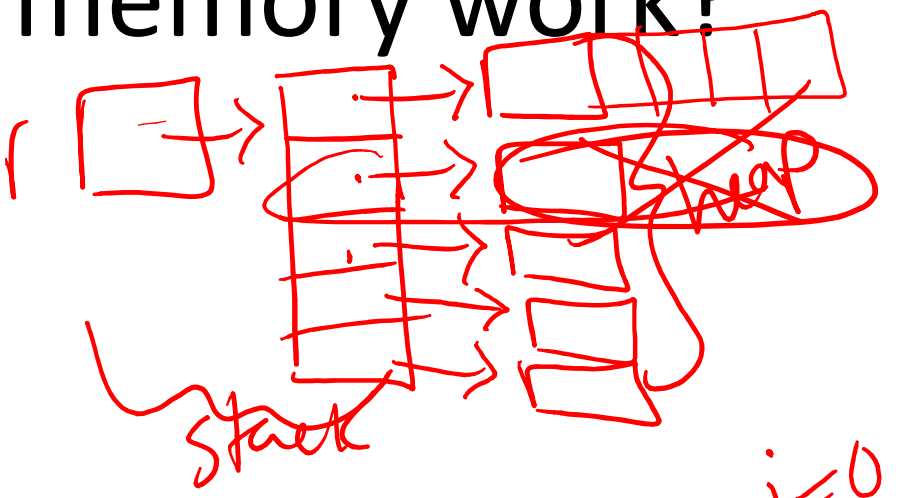
How does freeing memory work?

```
int *r[5], **s;

for(int i=0; i < 5; i++)
  r[i]=new int;
for(int i=0; i < 5; i++)
  delete r[i];

for(int i=0; i < 5; i++)
  r[i]=new int[5];
for(int i=0; i < 5; i++)
  delete [] r[i];

s=new int*[5];
for(int i=0; i < 5; i++)
  s[i]=new int[5];
for(int i=0; i < 5; i++)
  delete [] s[i];
delete [] s;
```



create

delete

i=0

delete [ ]

(i++)

r[i]