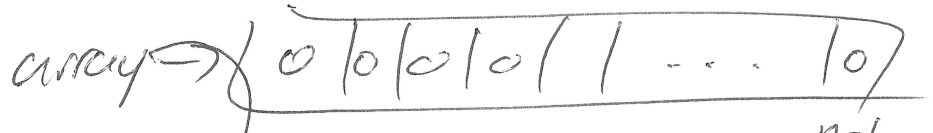
COP 3502 9/1/2021

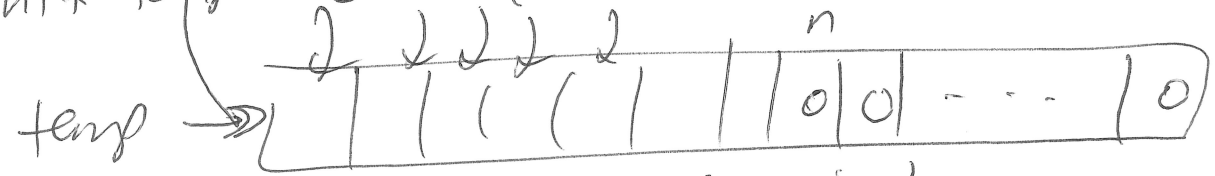
1. calloc, realloc
2. array of struct
3. array pt of struct
4. QUIZ

```
int* array = calloc(n, sizeof(int));
```



Double size manually

```
int* temp = calloc(2*n, sizeof(int));
```




```
for (int i=0; i<n; i++)
    temp[i] = array[i]
free(array);
array = temp
```

Switching these is catastrophic!

```
array = realloc(array, 2*n*sizeof(int));
```

↳ OR DO THIS

Addition Program (example w/ realloc, calloc)

Short Circuiting for arrays + ternary operator are very important for avoiding array out of bounds

lines 100, 101

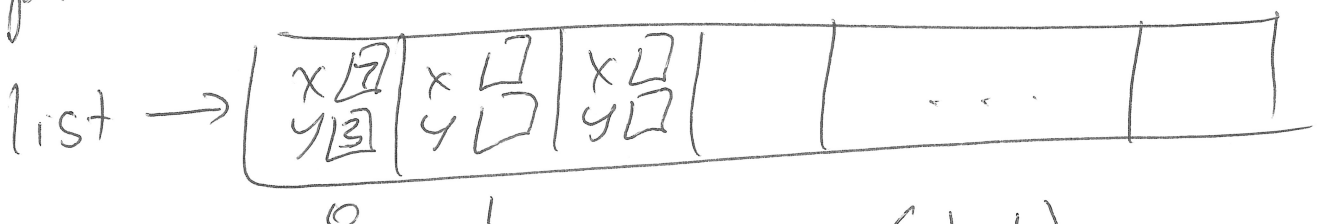
s.c. if (i < n && array[i] < low)

↑ check first

Array of Struct

pt

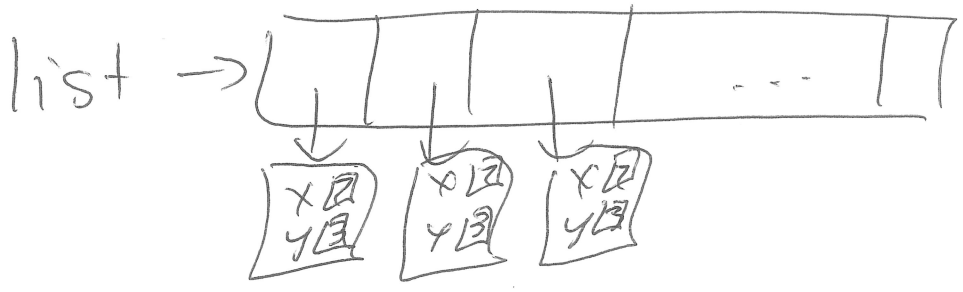
```
typedef struct pt {
    int x;
    int y;
} pt;

pt* list = calloc(10, sizeof(pt));
```




```
list[0].x = 7;
list[0].y = 8;
```

```
free(list);
```

Array of pt to Struct




```
pt** list = calloc(10, sizeof(pt*));
for (int i = 0; i<10; i++) {
    list[i] = malloc( sizeof(pt) );
    list[i]->x = 7 // (*list[i]).x
    list[i]->y = 3;
}
```

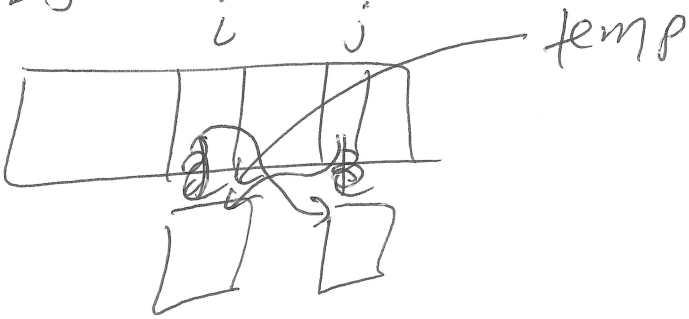
How to free

```
for (int i = 0; i<10; i++)
    free(list[i]);
free (list);
```

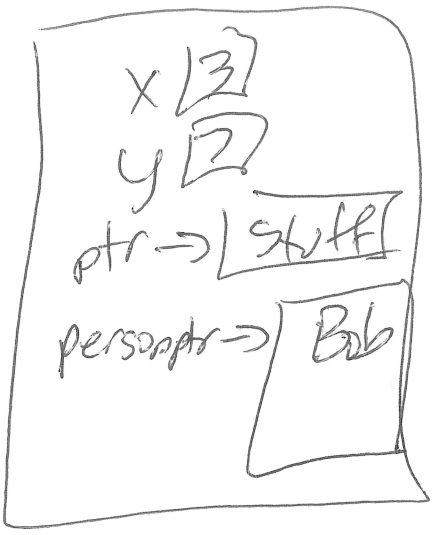
Why do I like this better?

Ans: To swap elements, I can do

```
pt* temp = list[i];
list[i] = list[j];
list[j] = temp;
```



Note: A struct can have anything



[ ] array

. struct

-> ptr struct

* ptr = The thing ptr is pointing to

& var = mem adr of var

QUIZ

Sec1 Paper - Pass out 9:25

8.5"x11" | Sheet of notes (typed, written front/back)

// malloc, calloc, realloc, free

Written Responses

Sec2 Paper - Pass out 12:25

8.5"x11" | Sheet of notes (typed, written front/back)

Written Responses