COP 3502 8/22/2025

1) Use of Ecists

2) Array of Strings 2 ways

3) Array of Struct

4) Array of Ptr to Struct

5) Smoothie Picture

→ Pre-typed

functions

malloc
calloc
realloc
free

Dictionary/List of words

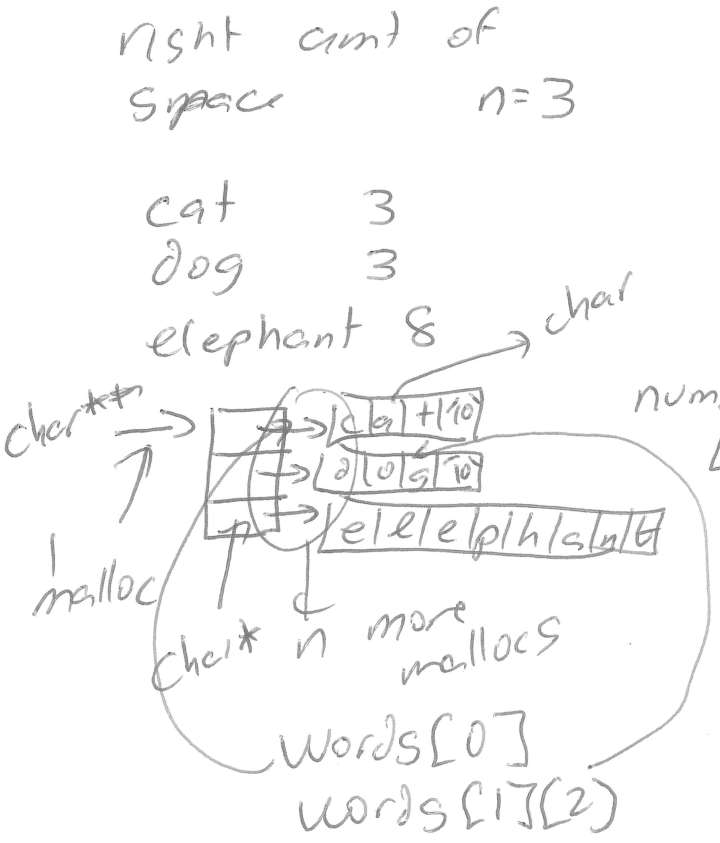
Strategy #1

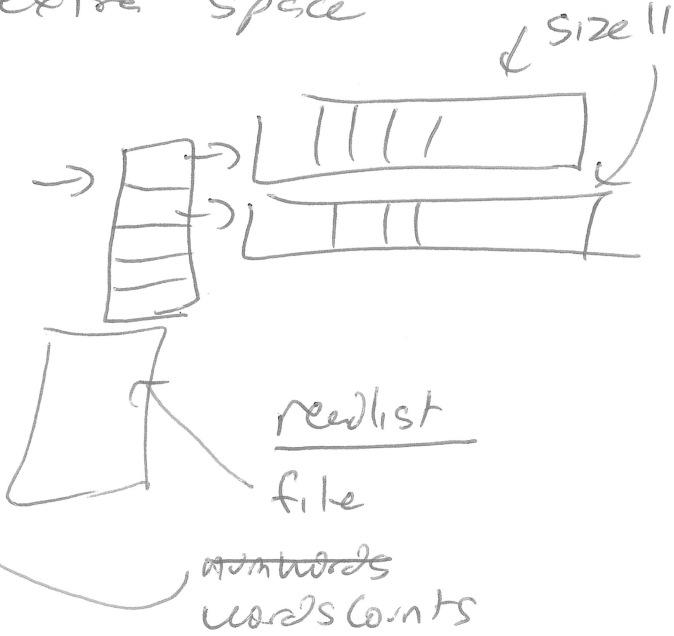
allocates just right amt of space n=3

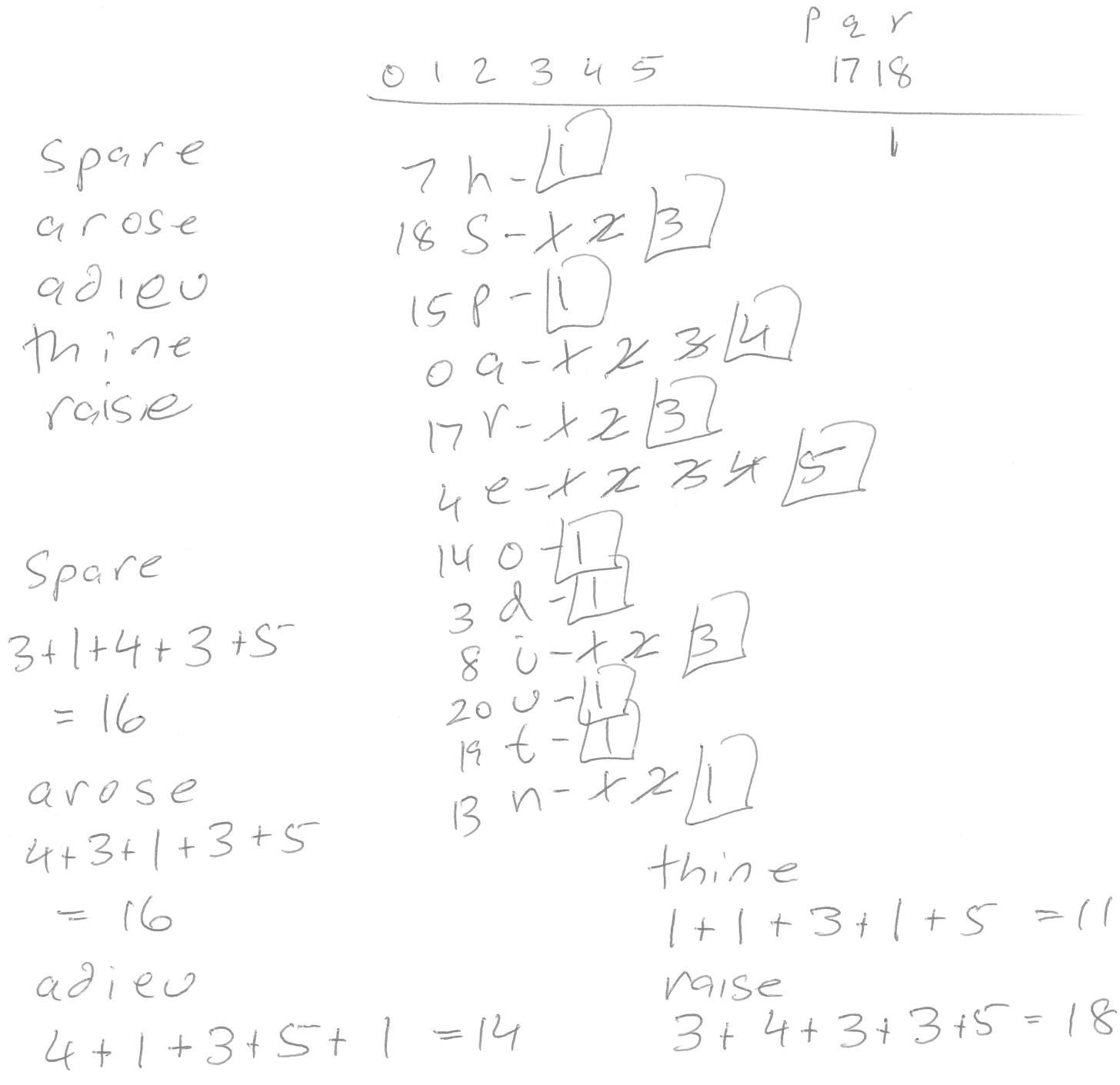
cat 3
dog 3
elephant 8

Strategy #2 max 10

bit easier does a little extra space






p q r
0 1 2 3 4 5
17 18
Spare
arose
adieu
thine
raise
7 h - 1
18 S - x x 3
15 p - 1
0 a - x x 3 4
17 r - x x 3
4 e - x x 3 4 5
14 0 - 1
3 d - 1
8 i - x x 3
20 u - 1
19 t - 1
13 n - x x 1
Spare
3+1+4+3+5
= 16
arose
4+3+1+3+5
= 16
adieu
4+1+3+5+1 = 14
thine
1+1+3+1+5 = 11
raise
3+4+3+3+5 = 18

3 meanings of *

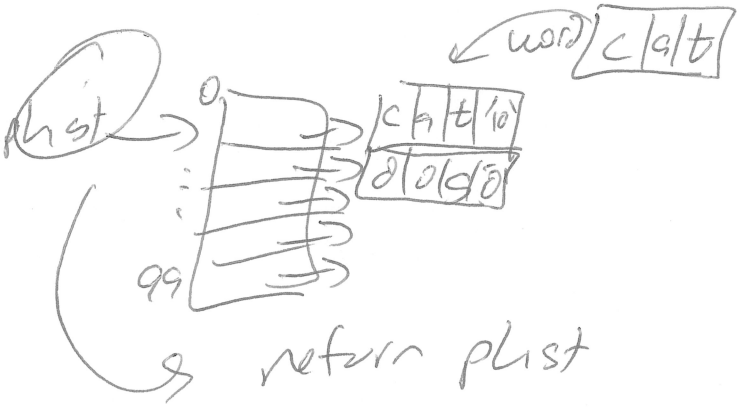
multiplication a * b

dereference a ptr * countWords

→ is the value in the variable countWords is pointing to

define a type char* word;

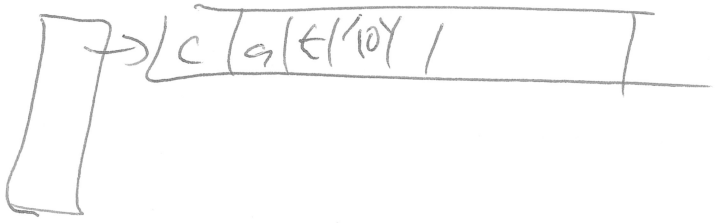
↳ word is of type char pointer



↳ return plist

In array alloc 2

1) malloc extra space



2) fscanf("%s", plist[i]);

When freeing memory for a nested structure free is reverse order so you don't lose access to any of the memory!

If you lose access ⇒ memory leak!

Array of Struct

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

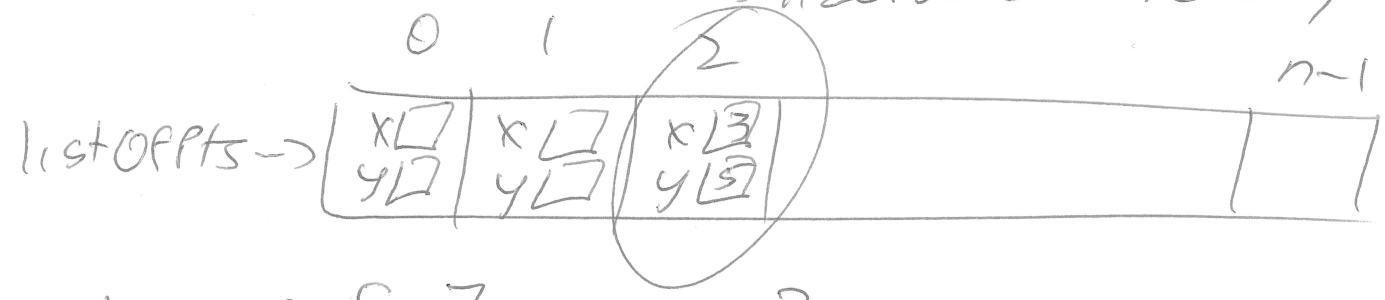
calloc

calloc(# items, ~~struc~~ Size of 1 item)

malloc (n * sizeof(pt))

```
pt * listOfPts = calloc (n, sizeof(pt));
```

// zeros out memory




```
listOfPts[2].x = 3
```

what type?

If the item to left is a struct, use a dot to access components

```
listOfPts[2].y = 5;
```