```
char *ptr[3] = {"one", "two", "three", "four", "five", "six"};
char **ptr1[3] = {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+3};
char **ptr2 = ptr1;

printf("%c", ++*++**ptr2++);
printf("%s", ++*--*++ptr2+3);
printf("%s", *ptr2[]+3);
printf("%s", --ptr2[-][-]+1);
```

(i) ++*++**ptr2++ → later

*ptr2 ⇒ (ptr+5)

*(ptr+5) = "six" → ptr to char

"six" → sptr sptr = sptr

sptr = sptr+1

"six" → sptr

*sptr ⇒ 'i'

⇒ ++'i' ⇒ 'j'

(ii) ptr2 = ptr1+1

++*--*++ptr2+3

ptr2 = &(ptr+4)

++ptr2

⇒ ptr2 = ptr2+1 (size of ptr-to-ptr-char)

= &(ptr+3)

~~*(&(ptr+3))~~

ptr+3 ⇒ &("four")

--(ptr+3)

⇒ ptr+3 − 1 (size of ptr-to-char)

⇒ ptr+2

*(ptr+2) ⇒ "three"

⇒ sptr

++sptr "three"

⇒ sptr → "three"

⇒ sptr+3

⇒ "three"

sptr

⇒ e

(iii) *ptr2[]+3

ptr2 = &(ptr+3)

((ptr2+1))

((&(ptr+2)))

*(ptr+2)

"three" → sptr (updated previously)

sptr+3

⇒ "e"

(iv) --ptr2[-][-]+1

ptr2 = &(ptr+3)

(ptr2−1) ⇒ ~~(&(ptr+4))~~

⇒ ptr+4

⇒ &("five")

*(&("five") − 1)

ptr+4 value

~~*(&("four"))~~

"four" → sptr

--sptr

"four" → sptr

sptr+1

"four" → sptr → sptr+1

"four"

```
char *ptr[3] = {"one", "two", "three", "four", "five", "six"};
char **ptr1[3] = {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+3};
char **ptr2 = ptr1;

printf("%c", ++*++**ptr2++);
printf("%s", ++*--*++ptr2+3);
printf("%s", *ptr2[]+3);
printf("%s", --ptr2[-][-]+1);
```

*ptr2
ptr+5
*(ptr+5)

ptr2 ⇒ ptr1+2
*(ptr1+3)
ptr+2

ptr ⇒ array of char pointers
ptr1 ⇒ array of pointers to char pointer
ptr2 ⇒ pointer to char pointer

ptr2 = ptr1

++*++**ptr2++ → ptr2 = ptr2+1 = ptr2+1 (size of ptr to char)

ptr2 is same as ptr1+1

ptr1[5]
&"six"
&"six"
'i'
'j'

++*--*++ptr2+3
ptr1+2
ptr+3
ptr+2
"three"
"three"
"three"

++ptr2 ⇒ ptr1+2

*ptr2[1]+3
ptr+2
ptr+5
%s
"six"

--ptr2[-1][-1] + 1
ptr+4
"four"
"four"
"four" ??

ptr2 ⇒ ptr1+2
*(ptr1+1)
ptr+4

ptr2 ⇒ ptr1+2
*(ptr2−1)
*(ptr1+1)
ptr+4
*(ptr+4−1)
*(ptr+3)

Lecture 8

```
char *ptr[3] = {"one", "two", "three", "four", "five", "six"};
char **ptr1[3] = {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+3};
char **ptr2 = ptr1;

printf("%c", ++*++**ptr2++);
printf("%s", ++*--*++ptr2+3);
printf("%s", *ptr2[]+3);
printf("%s", --ptr2[-][-]+1);
```

arr[][3] = {11, 12, 13, 14, 15, 16, 17, 18, 19}
0 1 2 3 4 5 6 7
1000 1004 1008 12 14 16 20 24 28

no of rows to skip
+
no of elems to skip in current row

base + [(5%3) + (5/3)] * size of every element

no of columns

1000 + [2+1] * 4 ⇒

11, 12, 13, 14, 15, 16, 17, 18, 19
row 1, row 2, row 3

⇒ 11, 14, 17, 12, 15, 18, 13, 16, 19
1000, 4, 8, 12, 16, 20, 24, 28, 32

&arr[2][1] = ?

no of cols. skipped (0th col in our case)
+
no of elems skipped in current col. no (2 in our case i.e. 12 & 13)

[(j * row3) + (i)] * size

⇒ [[(j−C3) * (R2−R3)] + (i−R3)] * size

ptr2 = ptr1

ptr2 = ptr1

ptr2 = ptr1

ptr2++
ptr2 = ptr2+1 (size of pointer to char)
= ptr1+1 (size of ptr to char)
= &(ptr+4)

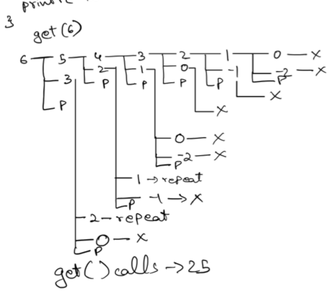
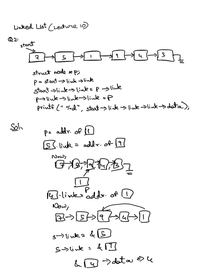
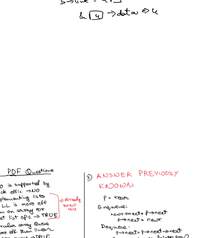
++ptr2
⇒ ptr2 = (ptr1+1)+1
= &(ptr+3)

*(&(ptr+3))
ptr+3

ptr2 = ptr1
ptr2 + 1 ⇒ ptr1 + 1 (size of ptr to char)
ptr2 = ptr1+1
++ptr2
ptr2 = ptr1+2
--*(ptr2)
--(ptr2) → ptr+3
⇒ ptr+3 = ptr+3−1
= ptr+2
*(ptr+2)
⇒ "three" *(ptr+2) + *(ptr+2)
⇒ hree

```
void get(int n){
    if(n<1) return;
    get(n-1);
    get(n-3);
    printf("%d", n);
}
```

get(6)



get() calls → 25