```
char *ptr[3]= {"one", "two", "three", "four", "five", "six"};
char **pptr1[3]= {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+0};
char ***pptr2= pptr1;

printf("%c", ++**++***pptr2++);
printf("%s", ++*--*++pptr2+3);
printf("%s", *pptr2[0]+3);
printf("%s", --pptr2[-1][-1]+1);
```

(i) ++**+***pptr2++ → later

*pptr2 ⇒ (ptr+3)

*(ptr+3) = "six" → ptr to char

"six" → sptr s=sptr

sptr = sptr+1

"six" → sptr

*sptr ⇒ 'i'

⇒ ++'i' ⇒ 'j'

(ii) pptr2 = pptr1+1

++*--*++pptr2+3

pptr2 = &(ptr+4)

++pptr2

⇒ pptr2 = pptr2+1 (size of ptr-to-ptr-char)

= &(ptr+3)

((pptr+3))

pptr+3 ⇒ &("four")

--(ptr+3)

⇒ pptr+3 - 1 (size of ptr-to-char)

⇒ pptr+2

*(ptr+2) ⇒ "three"

⇒ sptr

++sptr "three"

⇒ sptr

⇒ sptr+3

⇒ "three" sptr

⇒ e

(iii) *pptr2[1]+3

pptr2 = &(ptr+3)

((pptr2+1))

((&(ptr++)))

*(ptr+2)

"three" → sptr (updated previously)

sptr+3

⇒ "e"

(iv) --pptr2[-1][-1]+1

pptr2 = &(ptr+3)

*(pptr2-1) ⇒ *(&(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 **pptr1[3]= {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+0};
char **pptr2= pptr1;

printf("%c", ++**++***pptr2++);
printf("%s", ++*--*++pptr2+3);
printf("%s", *pptr2[0]+3);
printf("%s", --pptr2[-1][-1]+1);
```

*pptr2
pptr+5
*(ptr+5)

pptr2 ⇒ pptr1+2
*(pptr1+3)
pptr+2

pptr2 ⇒ pptr1+2
*(pptr2-1)
*(pptr1+1)
pptr+4
*(pptr+4-1)
*(pptr+3)

ptr ⇒ array of char pointers
pptr1 ⇒ array of pointers to char pointer
pptr2 ⇒ pointer to char pointer

pptr2 = pptr1

++**++***pptr2++ → pptr2 = pptr2+1 = pptr2+1 (size of ptr to char) ⇒ pptr1+1 (is same as)

pptr5
& "six"
& "six"
'i'
'j'

++pptr2 ⇒ pptr1+2

++**--*++pptr2+3
pptr1+2
pptr+3
pptr+2
"three"
"three"
"three"

*pptr2[1]+3
pptr+2
pptr+5
'i'
"six"

--pptr2[-1][-1]+1
pptr+4
"four"
"four"
"four" ??
"four"

Lecture 8

arr[][8] = {11, 12, 13, 14, 15, 16, 17, 18}, {19, 20, 21, 22, 23, 24, 25, 26}, {27, 28, 29, 30, 31, 32}

no of rows to skip

no of elems to skip in current row

base + [(5/8) + (5%8)] * size of every element

no of columns

1000 + [2+3] * 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][3] = ?

no of cols skipped (oth col in own case)
+
no of elems skipped in current col (2 in own case i.e. 12 & 5)

[(i * row) + (i)] * size

⇒ [(i-(5)) * (R.E-R.S)] + (i-R.S)] * size

```
char *ptr[3]= {"one", "two", "three", "four", "five", "six"};
char **pptr1[3]= {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+0};
char **pptr2= pptr1;

printf("%c", ++**++***pptr2++);
printf("%s", ++*--*++pptr2+3);
printf("%s", *pptr2[0]+3);
printf("%s", --pptr2[-1][-1]+1);
```

pptr2++,
pptr2 = pptr2+1 (size of pointer to char)
= pptr1+1 (size of ptr to char)
= &(ptr+4)

++pptr2
⇒ pptr2 = (pptr+1)+1
= &(ptr+3)
*(&(ptr+3))
ptr+3

pptr2 = pptr1

pptr2+1 ⇒ pptr1+1 (size of ptr to char)

pptr2 = pptr2+1

++pptr2

pptr2 = pptr1+2

--*(pptr2)

--(pptr+3)

⇒ pptr+3 = pptr+3-1
= pptr+2

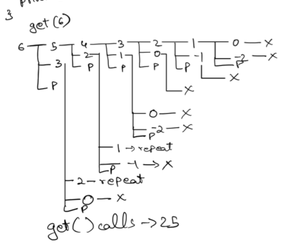
*(pptr+2)

⇒ "three" *(pptr+3) = *(pptr+2)+

⇒ hree

char *ptr[3]= {"one", "two", "three", "four", "five", "six"};
char **pptr1[3]= {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr+0};
char **pptr2= pptr1;

printf("%c", ++**++***pptr2++);
printf("%s", ++*--*++pptr2+3);
printf("%s", *pptr2[0]+3);
printf("%s", --pptr2[-1][-1]+1);

Read the question right - it is get(6)

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