int a [10]

→ array of size 10.

before runtime.

→ either wastage of memory

→ executes millions of time.

memory needs to allocate ?? runtime.

~~int a[10];~~ b[10], c[10] ... x[10];

int *a;

incr [10]

legacy code

COBOL

Y2K issue.

a = (int *) malloc (i * sizeof (int))

if [??!

free(a);

```
int *a;
(int*)
a = malloc(x * sizeof(int));
```

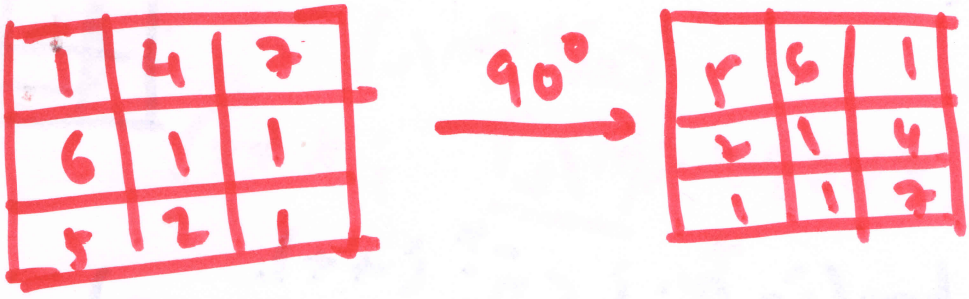
```
free(a);
```

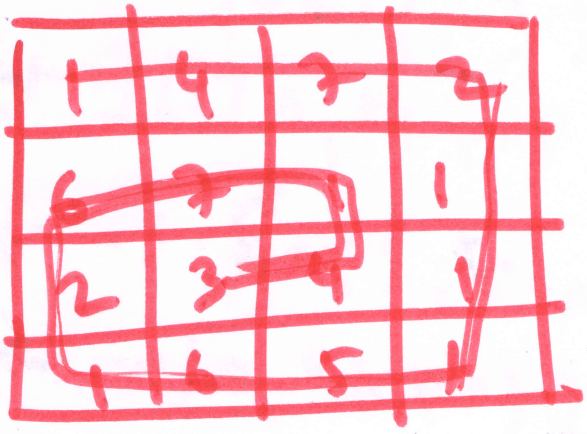
```
int mark[4][5];

int main()
{ int mark[40][50], row, column;
  scanf("%d%d", &row, &column);
for(i=0; i<row; i++)
  for(j=0; j<column; j++)
    scanf("%d", &mark[i][j]);

for(i=0; i<row; i++)
  { for(j=0; j<column; j++)
      printf("%d", mark[i][
    printf("\n")
```

column, row = 3

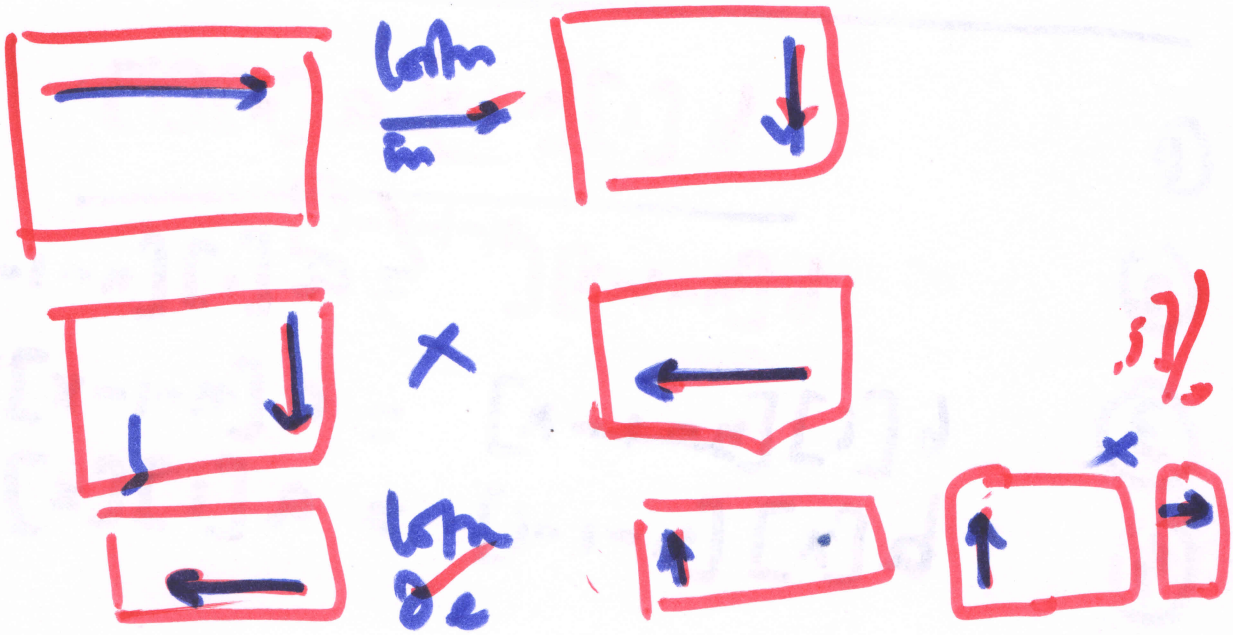
1	4	7
6	1	1
5	2	1

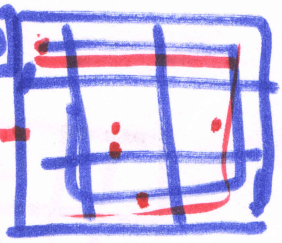
1 4 7
6 1 1
5 2 1





```
for(;;k++)
 { printf(
```





```
int a[40][40], b[40][40];
ceil(n/2)
for(x=0; x<⌈n/2⌉ ; x++)
{for(i=0; i<n; i++)
b[i][n-1-x] = a[x][i];                    ①
printf("%d", a[x][i]);
for(i=x+1; i<n-x; i++)
printf("%d", a[i][n-1-x]);                ②
for(i=n-2-x; i>=x; i--)
printf("%d", a[n-1-x][i]);                ③
for(i=n-2-x; i>x; i--)
printf("%d", a[i][x]);                    ④
}
```

---

① $b[i][n-1-x] = a[x][i]$

② $b[n-1-x][n-1-x-i] = a[i][n-1-x]$

③ $b[i][n-1-x] = a[n-1-x][i]$

④ $b[x][n-1-i] = a[i][x]$

$b[i][n-1-x] = a[x][i]$

$b[n-1-x][n-1-i] = a[i][n-1-x]$

$b[i][x] = a[n-1-x][i]$

$b[x][n-1-i] = a[i][x]$

0 ✓ 1

$i = n-2-x \geq 0$

$= n-1-(n-2)$

$i = 1$