

MEMORY MANAGEMENT

To store anything in our computer, we should have to allocate the memory first.

This memory allocation is conducted in two ways.

1. Static memory allocation.
2. Dynamic memory allocation.

In static memory allocation, the memory specified at compile/design time, based on the data type or array size. This type of memory management is called compile time memory management [**compiler indicates memory and O.S allocates the memory**].

In static memory allocation, the memory size is fixed at compile time and we can't

change this memory size at run time. It causes some times memory wastage / shortage.

To avoid this problem, the only solution is dynamic memory allocation.

In dynamic memory allocation, the memory is allocated at run time, based on the user input, instantly.

This type of memory management is called run time memory management.

To conduct dynamic memory allocation, we should have to use **pointers**.

In dynamic memory allocation the memory is allocated in **HEAP** area.

To manage the dynamic memory, we are using some predefined functions like

- malloc()
- calloc()
- realloc()
- free()

All these functions are available in **<alloc.h>**

malloc(), realloc(), calloc() functions are able to allocate the memory of **64KB**

Maximum at a time.

To allocate more than 64KB memory, use the functions

- farmalloc()
- farcalloc()
- farrealloc().

Note:

when we are working with dynamic memory allocation, we have to allocate the

memory for any data type. Due to this all these functions return datatype is **void ***, which is a generic type. Due to this we should have to provide **explicit type casting** for all these functions.

malloc()	calloc()
Block Memory allocation	Contiguous memory blocks allocation
Allocates memory in bytes form	Allocates memory in blocks form.
Initial values garbage	Initial values 0
One argument required	Two arguments required
Used for normal variables	Used for array type variables

Syntax:

```
void * malloc(bytes);
```

```
void * calloc(no of blocks, block_size);
```

free(): It is used to release the memory allocated by malloc(), calloc() and realloc().

Syntax: void free(pointer);

realloc(): It is used to extend the memory allocated by malloc() or calloc() at runtime. Working style is similar to malloc().

Syntax: void * realloc(oldptr, newsize);

free example:

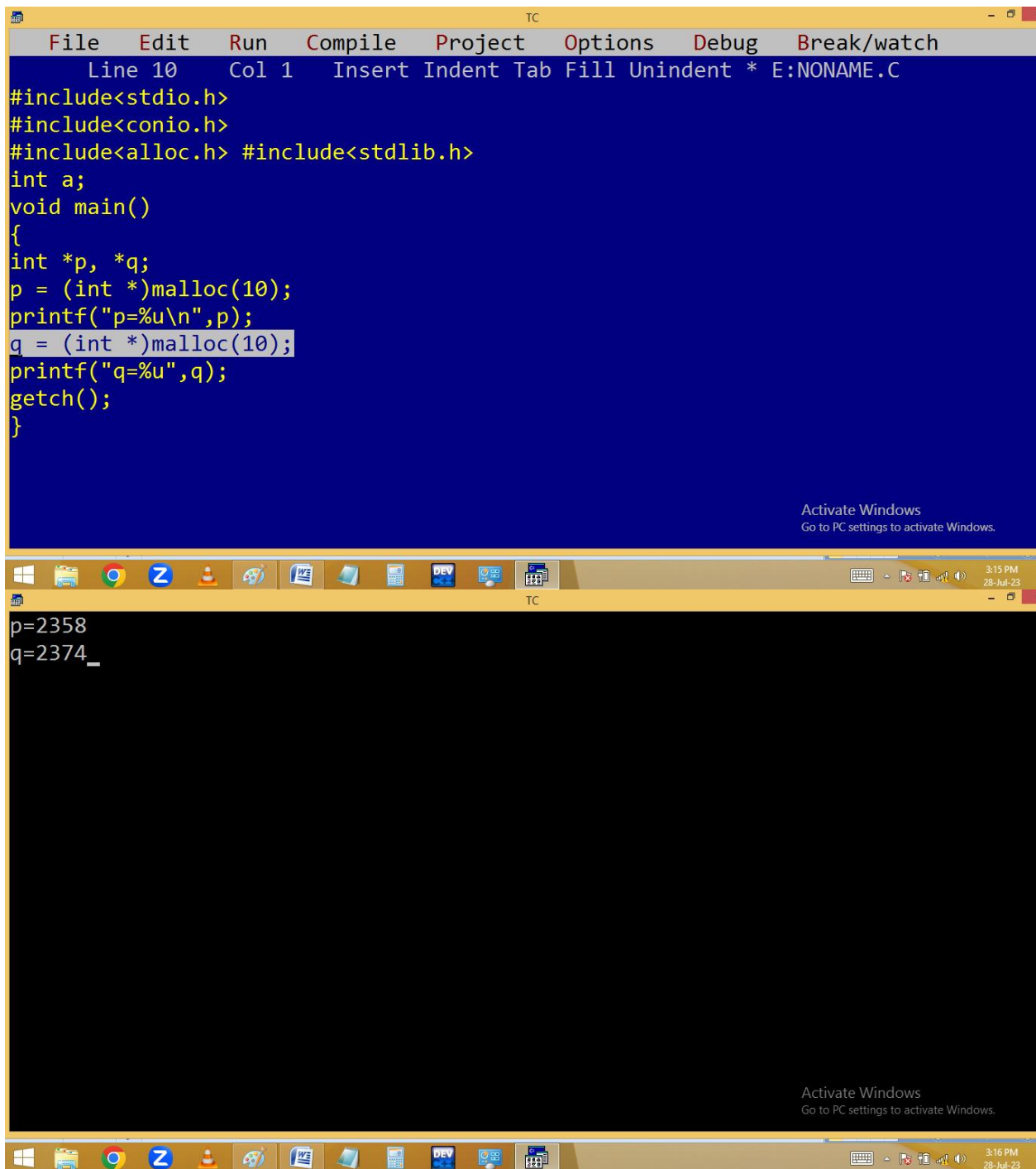
The image shows a Turbo C++ (TC) IDE window with a menu bar (File, Edit, Run, Compile, Project, Options, Debug, Break/watch) and a toolbar. The main editor area has a blue background and contains the following C code:

```
Line 10 Col 9 Insert Indent Tab Fill Unindent * E:NONAME.C
#include<stdio.h>
#include<conio.h>
#include<alloc.h> #include<stdlib.h>
int a;
void main()
{
int *p, *q;
p = (int *)malloc(10);
printf("p=%u\n",p);
free(p);
q = (int *)malloc(10);
printf("q=%u",q);
getch();
}
```

The output window at the bottom shows the execution results:

```
p=2358
q=2358
```

Both the IDE and the output window display a watermark: "Activate Windows Go to PC settings to activate Windows." The Windows taskbar at the bottom shows the time as 3:15 PM on 28-Jul-23.



The image shows a screenshot of the Turbo C++ (TC) IDE. The top window displays a C program with the following code:

```
File Edit Run Compile Project Options Debug Break/watch
Line 10 Col 1 Insert Indent Tab Fill Unindent * E:NONAME.C
#include<stdio.h>
#include<conio.h>
#include<alloc.h> #include<stdlib.h>
int a;
void main()
{
int *p, *q;
p = (int *)malloc(10);
printf("p=%u\n",p);
q = (int *)malloc(10);
printf("q=%u",q);
getch();
}
```

The bottom window shows the output of the program:

```
p=2358
q=2374_
```

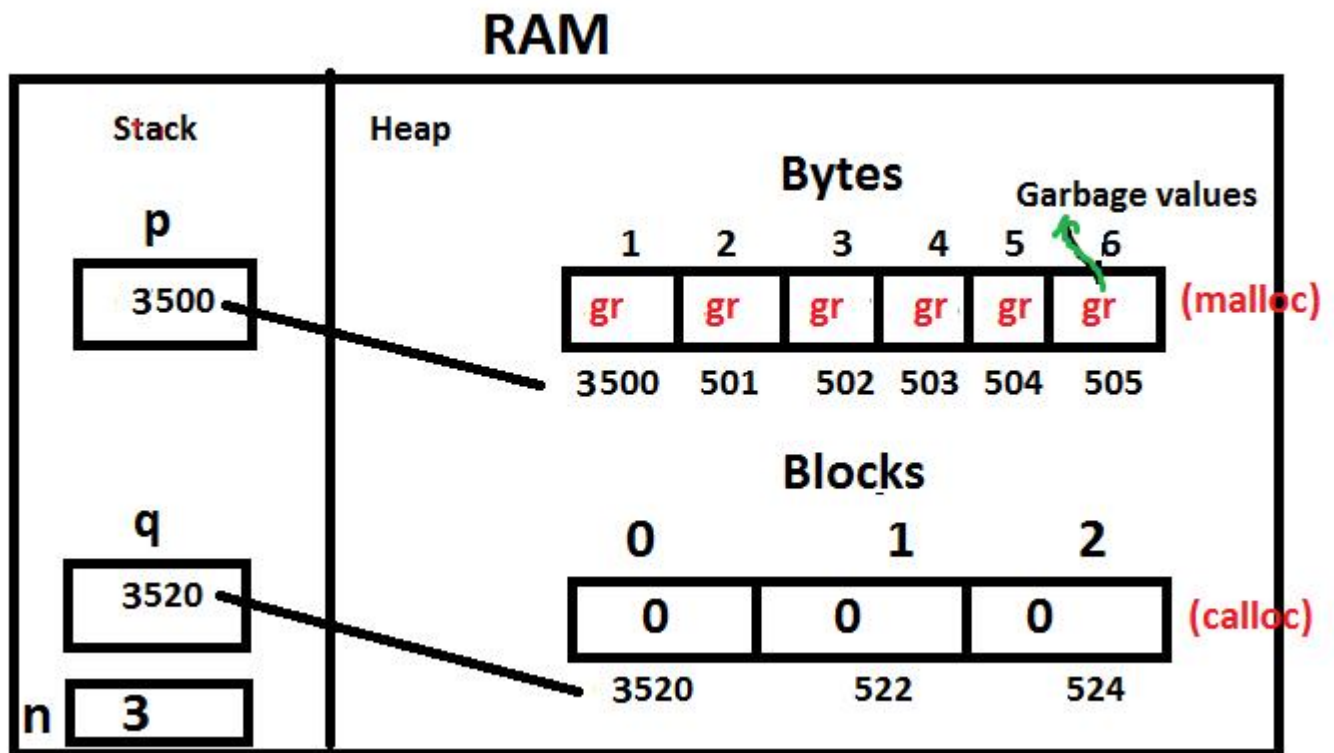
Both windows include a status bar at the bottom with the text "Activate Windows Go to PC settings to activate Windows." and a taskbar at the very bottom showing various application icons and the system clock (3:15 PM, 28-Jul-23).

allocating memory for 3 integers using malloc(), calloc().

```
int *p, *q, n=3;
```

```
p = (int *)malloc(n * sizeof(int));
```

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



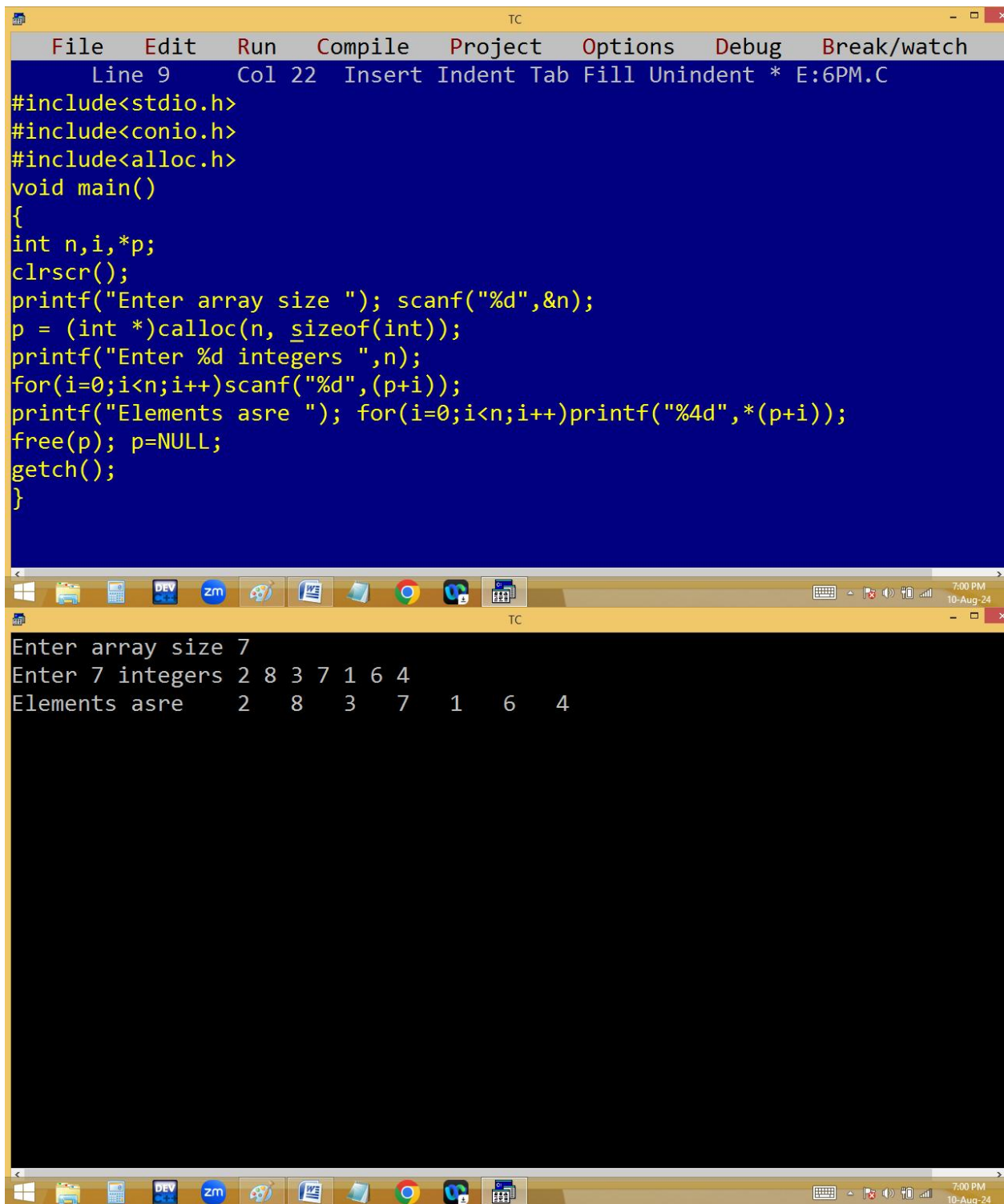

```
TC
File Edit Run Compile Project Options Debug Break/watch
Line 13 Col 17 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
void main()
{
int n,i,*p;
clrscr();
printf("Enter array size "); scanf("%d",&n);
p = (int *)malloc(n*sizeof(int));
printf("Enter %d integers ",n);
for(i=0;i<n;i++)scanf("%d",&p[i]);
printf("Elements asre "); for(i=0;i<n;i++)printf("%4d",p[i]);
free(p); p=NULL;_
getch();
}
```

```
TC
Enter array size 3
Enter 3 integers 100 9 15
Elements asre 100 9 15
```

```
TC
Enter array size 5
Enter 5 integers 2 0 1 7 5
Elements asre 2 0 1 7 5_
```

```
TC
File Edit Run Compile Project Options Debug Break/watch
Line 12 Col 63 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
void main()
{
int n,i,*p;
clrscr();
printf("Enter array size "); scanf("%d",&n);
p = (int *)malloc(n*sizeof(int));
printf("Enter %d integers ",n);
for(i=0;i<n;i++)scanf("%d",(p+i));
printf("Elements asre "); for(i=0;i<n;i++)printf("%4d",*(p+i));
free(p); p=NULL;
getch();
}
```

```
TC
Enter array size 4
Enter 4 integers 1 2 3 4
Elements asre 1 2 3 4_
```

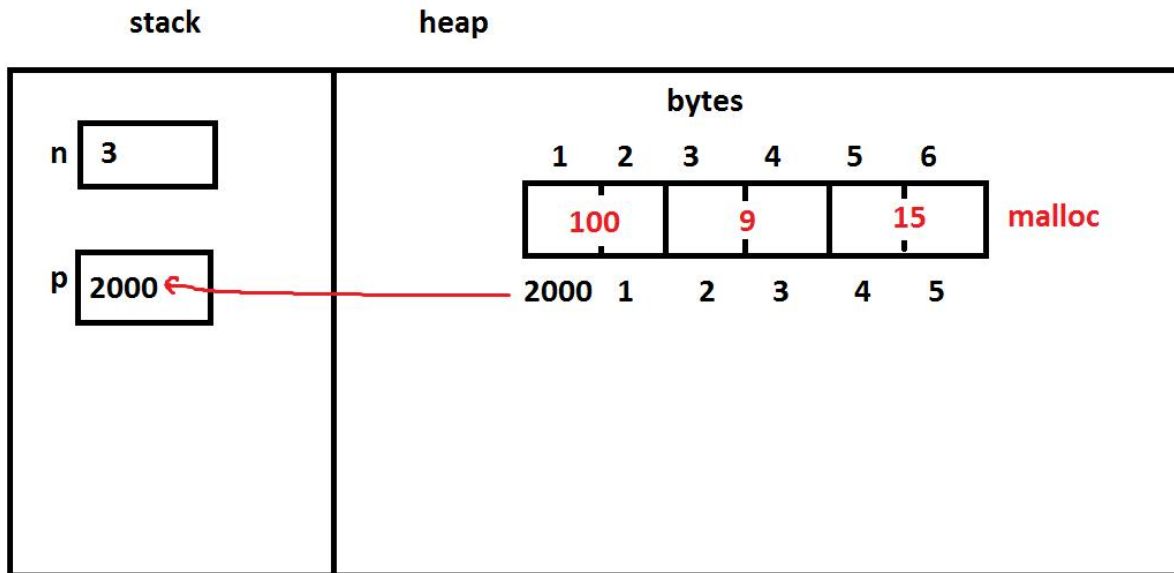


The image shows a screenshot of the Turbo C++ (TC) IDE. The top window displays the source code of a C program. The code includes headers for `stdio.h`, `conio.h`, and `alloc.h`. The `main` function starts by clearing the screen with `clrscr()`, then prompts the user to enter an array size. It uses `scanf` to read the size into `n`. A memory block is allocated using `calloc` to store the array. The user is then prompted to enter `n` integers, which are read into the array using a loop and `scanf`. Finally, the array elements are printed using `printf` in a formatted manner, and the program ends with `getch()`.

```
File Edit Run Compile Project Options Debug Break/watch
Line 9 Col 22 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
void main()
{
int n,i,*p;
clrscr();
printf("Enter array size "); scanf("%d",&n);
p = (int *)calloc(n, sizeof(int));
printf("Enter %d integers ",n);
for(i=0;i<n;i++)scanf("%d",(p+i));
printf("Elements asre "); for(i=0;i<n;i++)printf("%4d",*(p+i));
free(p); p=NULL;
getch();
}
```

The bottom window shows the execution output of the program. It displays the prompts and the user's input. The array size entered is 7, and the 7 integers entered are 2, 8, 3, 7, 1, 6, and 4. The output shows these elements printed in a formatted manner.

```
Enter array size 7
Enter 7 integers 2 8 3 7 1 6 4
Elements asre 2 8 3 7 1 6 4
```



Creating a dynamic multi dimensional array:

```
TC
Line 18 Col 9 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int **p,nr,nc,r,c;
clrscr();
printf("Enter no of rows and columns "); scanf("%d %d",&nr, &nc);
p = (int **)calloc(nr,sizeof(int));
for(r=0;r<nr;r++)p[r]=(int *)calloc(nc,sizeof(int));
printf("Enter %d integers\n",nr*nc);
for(r=0;r<nr;r++)for(c=0;c<nc;c++)scanf("%d",&p[r][c]);
puts("Elements");
for(r=0;r<nr;r++){for(c=0;c<nc;c++)printf("%4d",p[r][c]);
printf("\n");free(p[r]); p[r]=NULL;}
free(p);p=NULL;
getch();
}

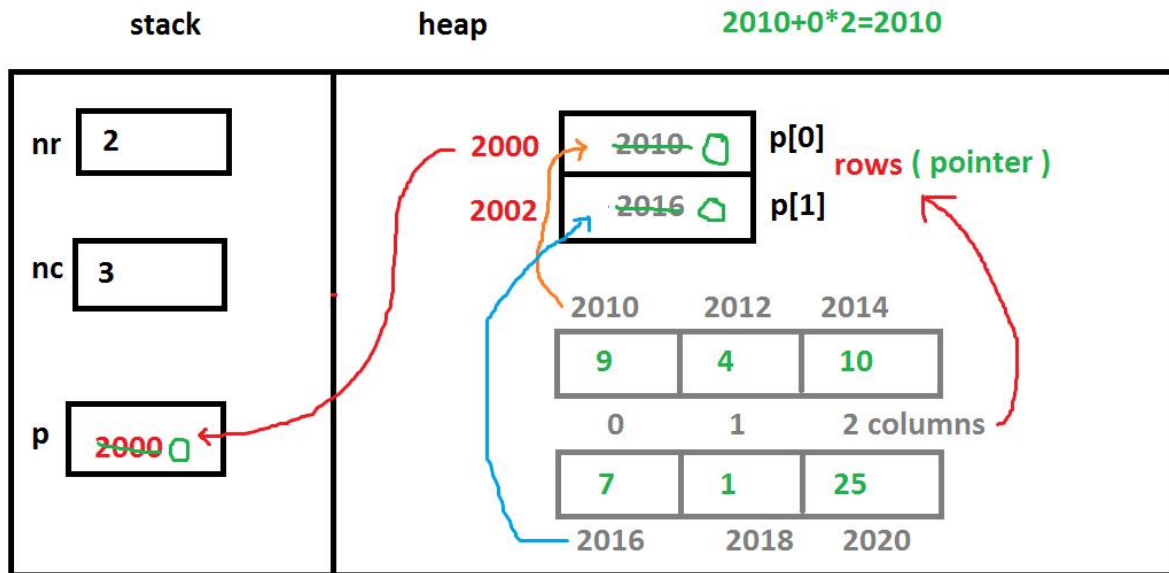
Enter no of rows and columns 2 3
Enter 6 integers
1 2 3 4 5 6
Elements
  1  2  3
  4  5  6

Page: 5 of 5 Words: 6 120% 7:23 PM 10-Aug-24
```

```
TC
Enter no of rows and columns 5 2
Enter 10 integers
1 3 2 7 4 9 4 0 8 5
Elements
1 3
2 7
4 9
4 0
8 5
```

```
TC
Line 18 Col 58 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int **p,nr,nc,r,c;
clrscr();
printf("Enter no of rows and columns "); scanf("%d %d",&nr, &nc);
p = (int **)calloc(nr,sizeof(int));
for(r=0;r<nr;r++)p[r]=(int *)calloc(nc,sizeof(int));
printf("Enter %d integers\n",nr*nc);
for(r=0;r<nr;r++)for(c=0;c<nc;c++)scanf("%d",*(p+r)+c);
puts("Elements");
for(r=0;r<nr;r++){for(c=0;c<nc;c++)printf("%4d",*(*(p+r)+c));
printf("\n");free(p[r]); p[r]=NULL;}
free(p);p=NULL;
getch();
}

Enter 4 integers
1 2 3 4
Elements
  1  2
  3  4
```

realloc() example:

```
TC
Line 18 Col 62 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int *p, s1, s2, i;
clrscr();
printf("Enter array size "); scanf("%d",&s1);
p = (int *) calloc( s1, sizeof(int));
printf("Enter %d integers ", s1);
for(i=0;i<s1;i++)scanf("%d",p+i);
printf("Enter no of cells to add ");scanf("%d",&s2);
p = (int *) realloc( p, (s1+s2)*sizeof(int));
printf("Enter %d integers ", s2);for(i=s1;i<s1+s2;i++)scanf("%d",p+i);
printf("Elements ");for(i=0;i<s1+s2;i++)printf("%4d",*(p+i));
free(p);p=NULL;
getch();
}

Enter 3 integers 8 3 9
Enter no of cells to add 2
Enter 2 integers 7 5
Elements      8   3   9   7   5
```

```
TC
Enter array size 4
Enter 4 integers 1 2 3 4
Enter no of cells to add 5
Enter 5 integers 0 8 4 8 7
Elements 1 2 3 4 0 8 4 8 7
```

