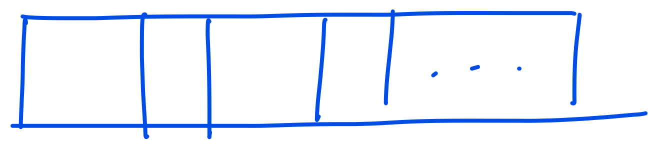
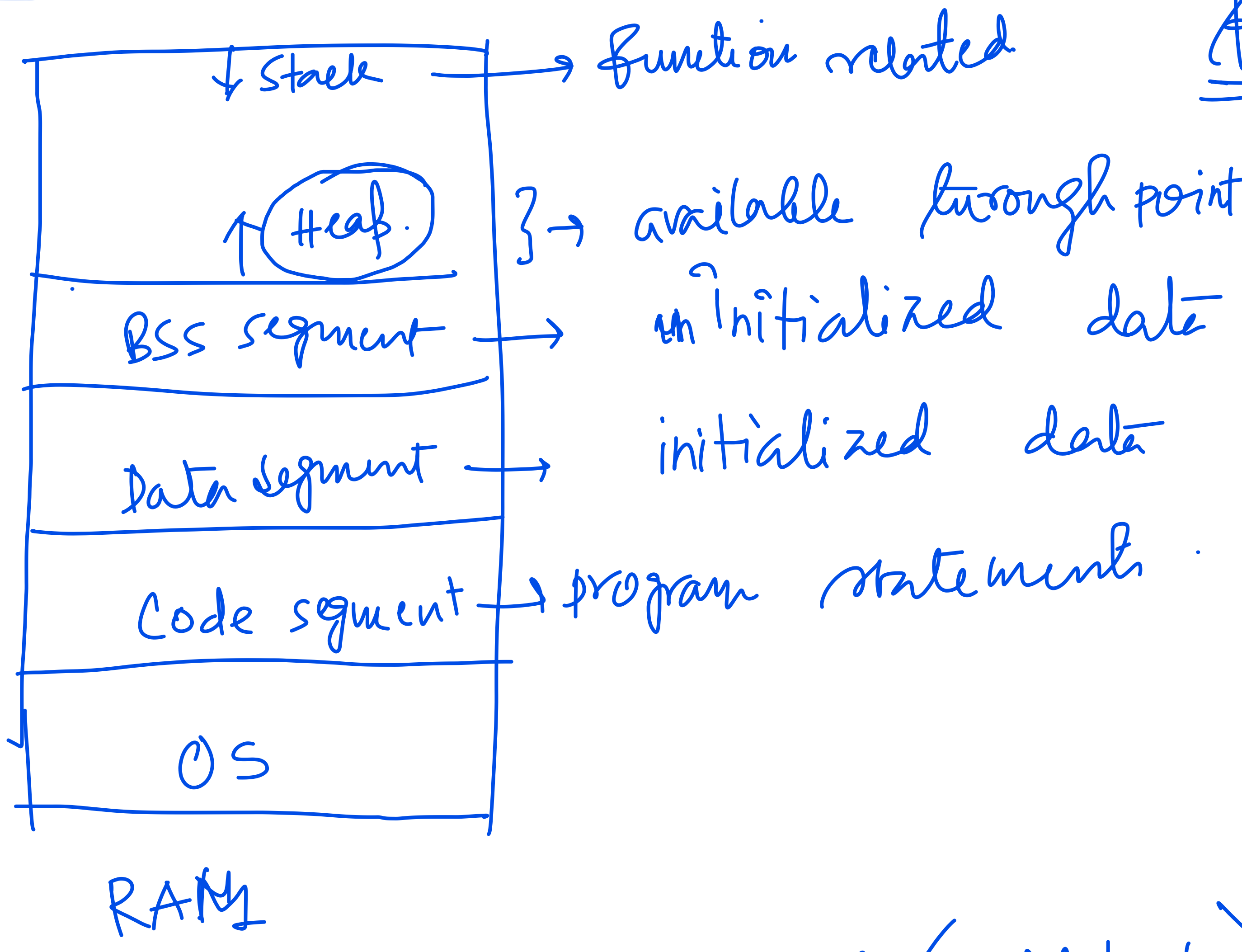
(Runtime) Dynamic memory allocation ??

int a[50];





function AR

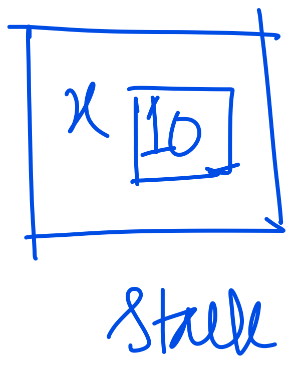
```
main()
{ int a;
  main();
}
```

o/p: stack overflow

malloc(), calloc(), realloc(), free() :- <stdlib.h>

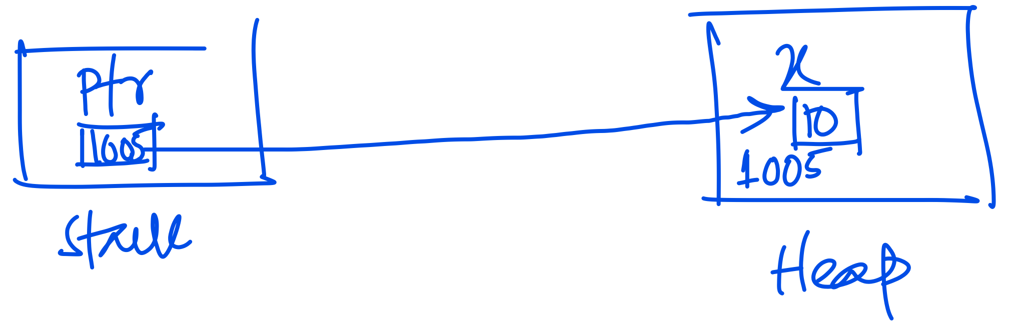
static (compilation time)

int x = 10; 4 bytes of memory is allocated at compilation time and the value 10 is stored. This mem is allocated on the "stack"



Static m/m allocation

* Static m/m allocation improves the performance of the program by faster access of the data.



* dynamic m/m allocation makes the program execution slower.

* While allocating the dynamic m/m the programmer should explicitly use functions like malloc(), calloc() and realloc() to allocate m/m. on heap.

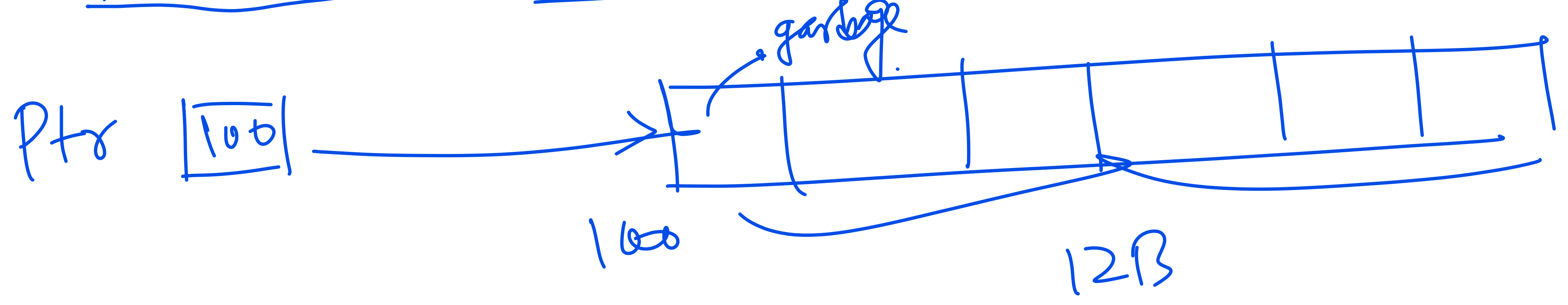
Once m/m usage is completed, the programmer should free the memory using free() function.

malloc

```
(void *) malloc (size_t size);
```

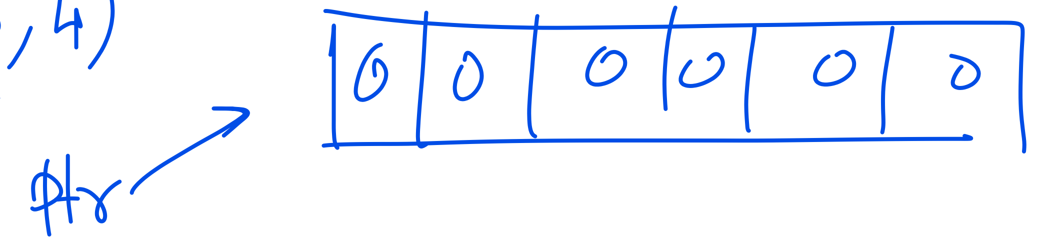
$\Downarrow$ argument specifies the no of bytes to be allocated.

```
int *ptr;
ptr = (int *) malloc (12);
```



A void pointer is a pointer that can be converted into any type of pointer.

```
Ptr = (int *) calloc (5,4)
```



String.

$\boxed{\text{'A'}}$ → character

"A" → String

A	\0

str[] = "debayan"

str

	d	e	b	a	y	a	n	\0
100	101	102	103	104	105	106	107	

$\boxed{\text{strlen ( )}}$:- library function

```
#include<stdio.h>
#include<string.h>
int main()
{
    char str[50];
    printf("enter a string");
    gets(str);
    printf("Length of the string is : %u", strlen(str));
        return 0;
}
```

Array version

```
unsigned int astrlen (char a[])
{   int i;
    for (i=0; a[i] != '\0' ; i++);
        return i;
}
```

pointer :-

```
unsigned int pastrlen (char *str)
{char *p=str;
  while (*p != '\0')
        p++;
     return p-str;
}
```