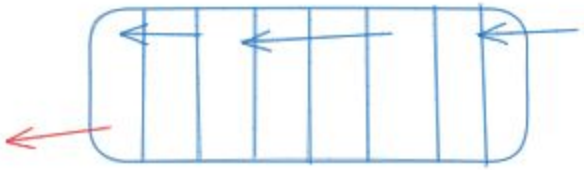
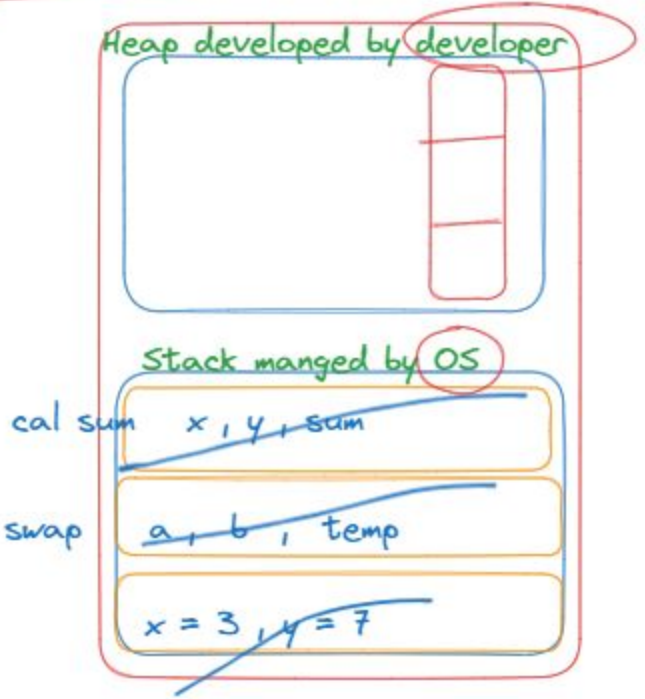
Dynamic Allocation

Stack : First in Last out

Queue : first in first out





malloc : heap لو عايز احجز مكان ف الميموري

pointer to void = malloc (20)

Casting : is transformaing values from one type to anthor

Implicit ضمني

when compiler automticaily
and convert from type to type

Explicit تصريحي

when compiler doesnot detect
automaticlly and the developer
needs to convert from type to type

```
int main() {
   int x = 8 , y = 3 ;
float z ;
 z = (float )x/y  ; //explicit x in this line float
printf("%f" , z) ;

}
```

```
int* ptr  = malloc (20) ;
struct Employee *emp ;
emp = malloc (sizeof(struct Employee )) ;
```

```
int main (){
   int emp_size ;
    printf("please enter size  : ") ;
scanf("%i" , &emp_size ) ;
 struct Employee * emp_ptr ;

// explicit
  emp_ptr = (Struct Employee ) malloc(emp_size * sizeof (struct Employee ))
//implicit
  emp_ptr = malloc(emp_size * sizeof (struct Employee)) ;
// fill
//emp_ptr++ ;
//rest
//display


 free ( emp_ptr ) ; // free allocation of address from the heap memory


}
```

Recursion

the process in which a function calls itself direct or indirect is called recursion and called recursive function ..

direct

فانكشن بتنادي نفسها بنفسها

indirect

الفانكشن بيحصل ليها نداء من فانكشن تانيه

```
void fun (int n ) {
    if (n== 1 ) {  // base case
return ;
    }
fun(n-1 ) ;

}
```

base case n==1 ;
statment fun(n-1) ;

fun (5)
↓
fun (4)
↓
fun(3)
↓
fun(2)
↓
fun(1)

ex get the sum to number from 1 to n

1	2	3	4	5	6	7	8	9
1	3	6	10	15	21	28	36	45

$f(n) = n + f(n-1)$;

n=5 ;

$f(5) = n + f(4)$;
$f(4) = n + f(3)$;
$f(3) = n + f(2)$;
$f(2) = n + f(1)$;
$f(1) = 1$ // base case

$f(1) = 1$;
$f(2) = 2 + 1 = 3$
$f(3) = 3 + 3 = 6$
$f(4) = 4 + 6 = 10$
$f(5) = 5 + 10 = 15$

```
int sum (int n ) {
  if (n==1 ) {
    return 1 ;
  }
  return n+ sum (n-1) ;
}
```

Calc power ;

$(2^4) = 2 * \boxed{2 * 2 * 2} = 16$

$(2^4) = 2 * (2^3)$

$(2^3) = 2 * (2^2)$

$(2^2) = 2 * (2^1)$

$(2^1) = 2 * (2^0)$

$(2^0) = 1$

$f(b^x) = b * (b^{(x-1)})$;