**1.**

\*(\*(M+1)+2)

**2.**

- Parametro ricorsivo: n

- O(n) = 2(O(n) = O(n)

- Decresce in base alla chiamata, la prima di 1 e la seconda di 3.

- Fattore minimo di decrescita: 1 <= n <= n+max

**3.**

Si sposta sul primo pezzo e poi va al secondo elemento = 5

**4.**

int \*p;

\*q = p; // primo elemento

\*r = q;

\*\*\*r = valore puntato da un puntatore di un puntatori

\*\*r + (i) = \*\*\*r

In memoria: [p, q, r]

int y = 20

\*p = y

Se modifichi “q” che “r” e altri; si ripercuote su tutti

\*puntatore = “cella di memoria”

int \*p = 42

// oppure

int \*p = &x;

\*\* = punti al primo elemento di uno \*

\*\*\* = punti al primo elemento di uno \*\*

**5.**

a.

[1, 2, 3, 4] = classico con int (4 B)

La funzione ritorna:

[null, null, null, null]

Ti stai riferendo a una variabile locale che viene deallocata alla fine della funzione createDanglingPointer.

Le variabili locali non salvo valore, invece i puntatori/riferimenti servono proprio a quello.

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

c.

local=42

return &local

if(local)

int \*p = local

return p;

**6.**

p 🡪 [10][][][] (4 B)

q = p 🡪 [q/p = 10][][][]

free 🡪 p = [][][][]

\*q = 20; 🡪

a. “q” è un dangling pointer

b.

Memory leak = accedi a un pezzo

Undefined behavior = comportamento non definito dallo standard del C

Se è indefinito, potrebbe succedere casini non coperti dallo standard (roba che non sia come gestire)

Caso main: return 0;

Accesso a nullptr: return -173627;

c.

Prima di fare free, dovremmo mettere p a NULL oppure q a NULL.

**7.**

9 = spostamento a una sola dimensione

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente