



PROG 1 • ESERCIZI D'ESAME - ALLOCAZIONE MEM

2.

```
#include <stdio.h>
#define DIM (size_t) (3)

int ric(size_t lenA, int a[], int i);

int main(void) {
    int v[DIM] = {10,5,1};
    int r = ric(DIM,v,0); // A
}

int ric(size_t lenA, int a[], int i) {
{
    if (i < lenA) {
        int n = a[i];
        a[i] = 0;
        return n + ric(lenA,a,i+1); // B
    }
    else {
        return 0;
    }
}
}
```

a

Indicare il valore di $v[2]$ prima della disallocazione della funzione main, cioè appena dopo la linea etichettata con la lettera

Scegli...
7
1
11
5
0
6

a

Indicare il valore restituito al termine della chiamata della funzione ric nel frame in cui $i == 2$

Indicare il valore restituito al termine della chiamata della funzione ric nel frame in cui $i == 1$

Indicare il valore di n prima di eseguire la linea etichettata con la lettera B nel frame della funzione ric, in cui $i == 1$

Indicare il valore di $v[2]$ prima della disallocazione della funzione main, cioè appena dopo la linea etichettata con la lettera

b

Indicare il valore restituito al termine della chiamata della funzione ric nel frame in cui $i == 2$

Indicare il valore restituito al termine della chiamata della funzione ric nel frame in cui $i == 1$

Indicare il valore di n prima di eseguire la linea etichettata con la lettera B nel frame della funzione ric, in cui $i == 1$

Indicare il valore di $v[2]$ prima della disallocazione della funzione main, cioè appena dopo la linea etichettata con la lettera

Scegli...
7
1
11
5
0
6

Terminati

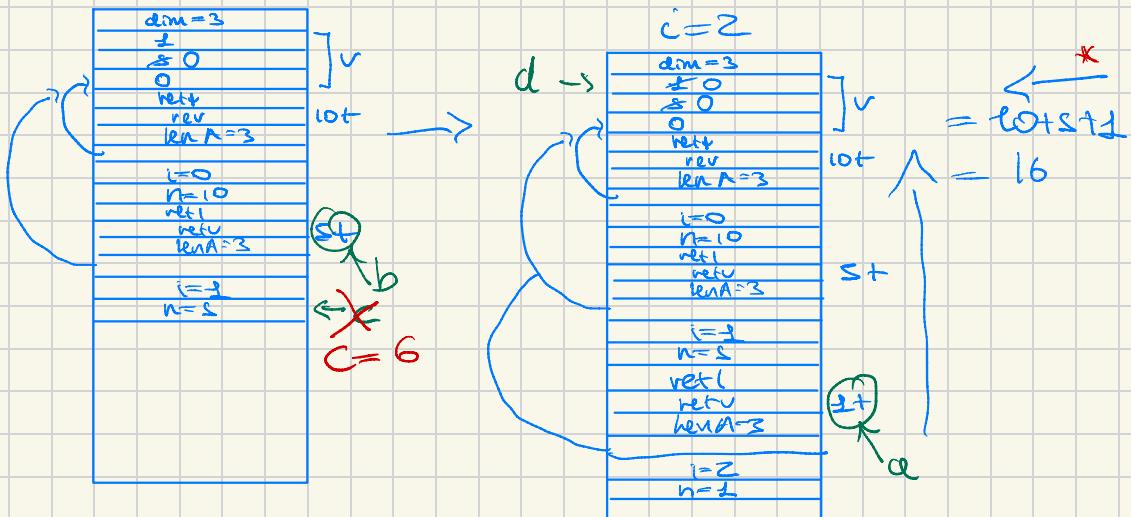
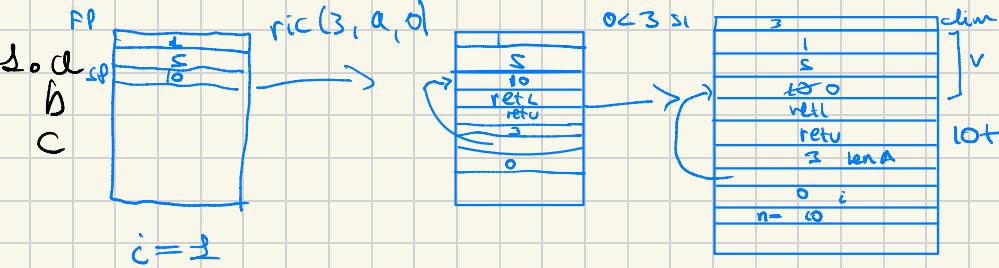
c

Indicare il valore di n prima di eseguire la linea etichettata con la lettera B nel frame della funzione ric, in cui $i == 1$

Indicare il valore di $v[2]$ prima della disallocazione della funzione main, cioè appena dopo la linea etichettata con la lettera

Scegli...
7
1
11
5
0
6

Slides presentate a lezione: Regole



PROG 1 • ESEMPI DI SAMM - CORR. ASS: 1

Dato il seguente sorgente C, usando il ragionamento backward, scegliere dai menu a tendina:

1. il predicato da asserire in ogni "assert";
2. la pre-condizione;

affinché la verità della pre-condizione implichia la verità della post-condizione.

```
void main (void)
/* pre-condizione : [ ] */
{
    post-condizione:  $33 \leq r \&& r \leq 39$ 
    /*
    void main (void)
    {
        assert ([ ]); // (WP)
        r = r + 2;
        assert ([ ]);
        r = r * 3;
        assert ([ ]);
    }
}
```

1.a

```
void main (void)
/* pre-condizione
   post-condizione
   */
void main (void)
```

1.b

```
void main (void)
{
    assert ([ ]); // (WP)
    r = r;
    assert ([ ]);
    r = r * 3;
}
```

1.c

```
r = r + 2;
assert ([ ]);
r = r;
assert ([ ]);
r = r * 3;
```

1.d

```
assert ([ ]);
33 <= r && r <= 39
34 <= r && r <= 38
32 <= r && r <= 38
```

1.a

$$\begin{aligned} 33 &\leq r'' + 2 \quad \& \quad r'' + 2 \leq 39 \rightarrow r' = r'' + 3 \quad \& \quad r'' = r' - 2 \\ r &= r + 2 \rightarrow 33 \leq r' * 3 \quad \& \quad r' * 3 \leq 39 \rightarrow r = r' * 3 \\ r &= r * 3 \end{aligned}$$

$\cancel{33 \leq r \& r \leq 39} \leftarrow 1.d$

POST: $33 \leq r \& r \leq 39$

$\cancel{2^*}$

$$33 \leq r' * 3$$

$$\frac{11}{3} \leq r'$$

$$\frac{1}{3} \leq \frac{39}{3}$$

$$r' \leq 13$$

$$\cancel{1.a} \quad (r'' + 2) * 3 \leq 39$$

$$1.b \rightarrow 3r'' + 6 \leq 39$$

$$\frac{3r''}{3} \leq \frac{33}{3}$$

$$r'' \leq 11$$

$\cancel{1.c}$

PROG-3 - Esercizi disegni - 2 Mecc. OR 1 4 50 ~~AP~~

//Si discopri il modello della genesi del cognato procreante:

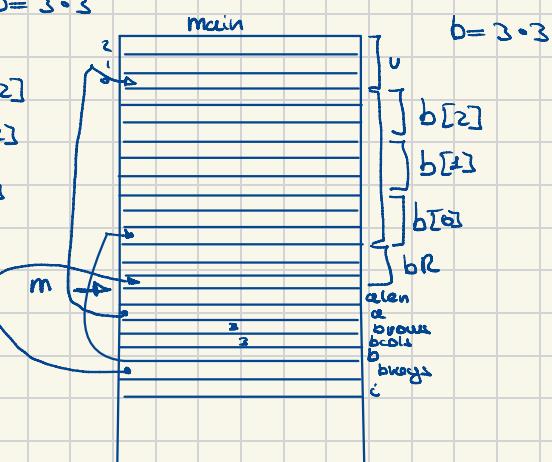
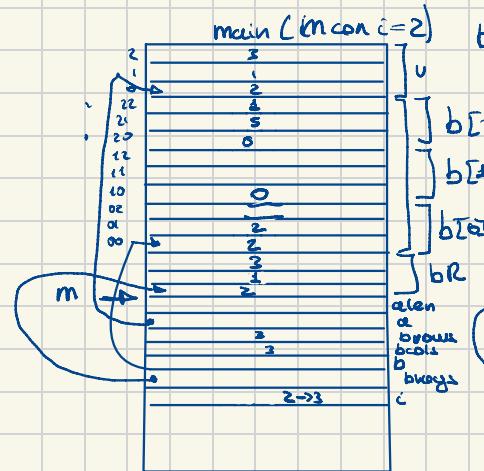
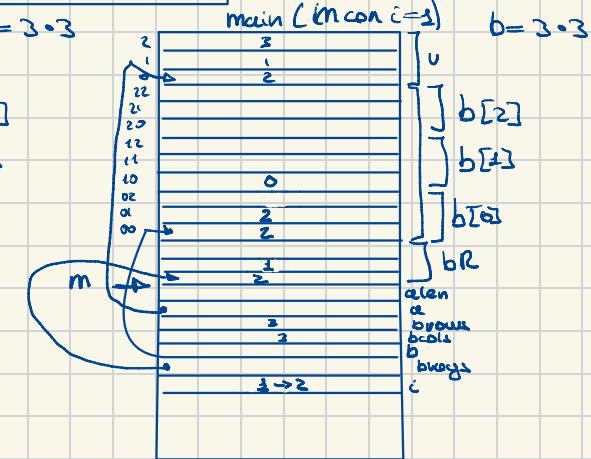
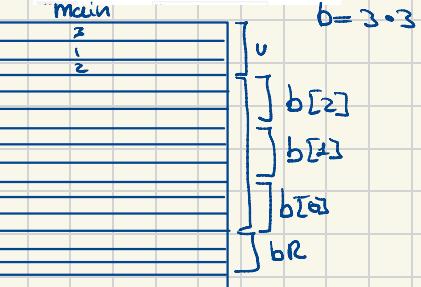
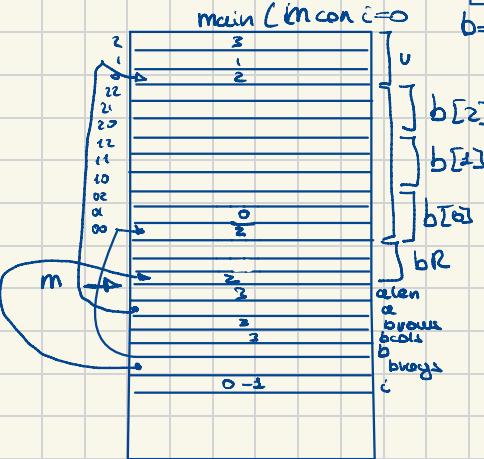
```

#include <stdio.h>
#define R (size_x_1) (3)
#define C (size_y_1) (1)
#define max (size_x_1, size_y_1)
#define min (size_x_1, size_y_1)

int main (void) {
    int R[3] = {2,3,1};
    int C[R[C]];
    size_t bRows;
    size_t bCols;
    m(R,a,B,C,bR,B,0); // A
}

void m(size_x_1, size_y_1, int a[], size_t bRows, size_y_1 bCols, int b[R][bRows], int i)
{
    if (i < size_x_1) {
        b[R[i]] = a[i];
        for (int k = 0; k < bRows; k++) {
            if (k % 2 == 0) {
                b[R[i]][k] = a[i];
            }
            else {
                b[R[i]][k] = a[i];
            }
        }
        m(R,a,bRows,bCols,bR,i+1); // B
    }
}

```



PROG-2 • ESERCIZI D'ESAME 3 Marzo

//Si disegni il modello della memoria del seguente programma:

```
#define DIM 3
void m(int a[], int x[], int i);
int main(void){
    int a[DIM];
    a[DIM,a,0]; // (A)
}
void m(int a[], int x[], int i){
    if (i < a.length){
        int x = a[i];
        m(x,a,i+1); // (B)
        a[i] = (i+1) * x;
    }
}
```

- 1 Indicare il valore di $a[2]$ immediatamente prima della disalloscrazione del frame della funzione m , quando $i == 0$
 Indicare il numero di frame dislocati (escluse il main) che hanno scritto nell'array x , immediatamente prima della disalloscrazione del frame della funzione m , quando $i == 1$
 Indicare il valore di $a[1]$ immediatamente prima della disalloscrazione del frame della funzione m , quando $i == 2$



- 3 Indicare il valore di $a[2]$ immediatamente prima della disalloscrazione del frame della funzione m , quando $i == 2$
 Indicare il valore di $a[1]$ immediatamente prima della disalloscrazione del frame del main



- 2 Indicare il numero di frame dislocati (escluse il main) che hanno scritto nell'array x , immediatamente prima della disalloscrazione del frame della funzione m , quando $i == 2$
 Indicare il valore di $a[1]$ immediatamente prima della disalloscrazione del frame della funzione m , quando $i == 2$



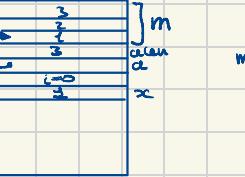
- 4 Indicare il valore di $a[1]$ immediatamente prima della disalloscrazione del frame della funzione m

4 Main, 02

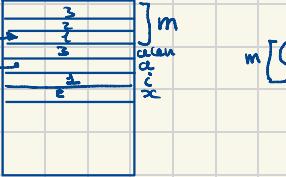
2 ora

4 NO = 3

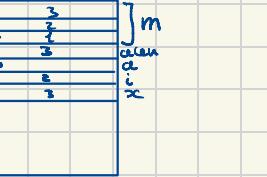
$m(i=0)$ Prima di ricors.



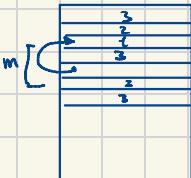
$m(i=1)$ Prima di ricors.



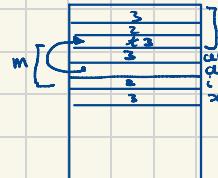
$m(i=2)$ Prima di ricors.



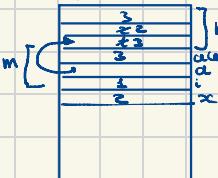
$m(i=3)$ Prima di ricors.



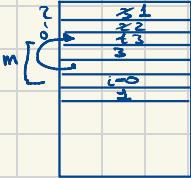
$m(i=2)$ Dopo Ricors.



$m(i=1)$ Dopo Ricors.



$m(i=0)$ Dopo di ricors.



$$a[3 - (2+1)] = a[0]$$

$$a[2 - (2+1)] = a[0]$$

$$a[3 - (1+1)] - a[1]$$

2 OK

PROG 1 • ESERCIZI D'ESAME 4 - Mem

3 ou

//Disegni il modello della memoria del seguente programma:

#define DIM 2

void m(int lenX, int x[], int i);

int main() {
 int x[4] = {1,2};
 m(DIM,x,0); // A
}void m(int lenX, int x[], int i) {
 if (i < lenX) {
 i++;
 m(lenX,x,i); // B
 }
}

1)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Indicare il valore di x[1] immediatamente prima della disallocazione del frame della funzione m.

Indicare il valore di x[0] immediatamente prima della disallocazione del frame della funzione m, quando i == 2

Quindi il punto A parte nel codice chiamato dopo la disallocazione del frame della funzione chiamata in questo i == 1

2)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Indicare il valore di x[0] immediatamente prima della disallocazione del frame della funzione m.

Indicare il valore di x[1] immediatamente prima della disallocazione del frame della funzione m, quando i == 2

Quindi il punto B parte nel codice chiamato dopo la disallocazione del frame della funzione chiamata in questo i == 1

3)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Indicare il valore di x[1] immediatamente prima della disallocazione del frame della funzione m.

Indicare il valore di x[0] immediatamente prima della disallocazione del frame della funzione m, quando i == 2

Quindi il punto C parte nel codice chiamato dopo la disallocazione del frame della funzione chiamata in questo i == 1

4)

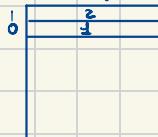
Indicare il numero di volte in cui la funzione m è stata chiamata.

Indicare il valore di x[1] immediatamente prima della disallocazione del frame della funzione m.

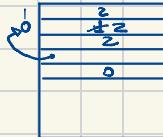
Indicare il valore di x[0] immediatamente prima della disallocazione del frame della funzione m, quando i == 2

Quindi il punto D parte nel codice chiamato dopo la disallocazione del frame della funzione chiamata in questo i == 1

main



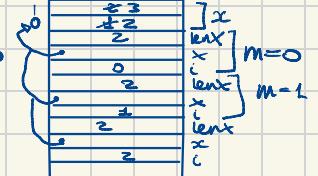
m(i=0)



m(i=1)



m(i=2) (Dis. main)



PROG 1 • ESERCIZI D'ESAME 5 - Mem

2 ou

//Disegni il modello della memoria del seguente programma:

#define width 10

#define height 5

#define ROWS (width * height)

#define COLS (height * width)

void a(int l, size_t size_x, size_t size_y, size_t aCols, bool aFlags[], size_t aSize, int aStep[aSize]);

int main(void)

{
 int l, size_x, size_y, aCols;

size_x = aStep[0];

for (l = 0; l < height; l++) {

aFlags[l] = 1;

aCols = width;

aStep[l] = aCols;

}

}

}

void a(int l, size_t size_x, size_t size_y, size_t aCols, bool aFlags[], size_t aSize, int aStep[aSize]) {

aFlags[l] = 0;

for (int i = 1; i < l; i++) {

aFlags[i] = 1;

aStep[i] = 0;

}

}

}

1)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Quanti valori false sono presenti e immediatamente prima della disallocazione del frame della funzione m?

Indicare il valore di aFlags[1] immediatamente prima della disallocazione frame della funzione m.

Quanti valori true sono presenti e immediatamente prima della disallocazione del frame della funzione m?

2)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Quanti valori false sono presenti e immediatamente prima della disallocazione frame della funzione m?

Indicare il valore di aFlags[1] immediatamente prima della disallocazione frame della funzione m.

Quanti valori true sono presenti e immediatamente prima della disallocazione frame della funzione m?

2 ou

3)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Quanti valori false sono presenti e immediatamente prima della disallocazione del frame della funzione m?

Indicare il valore di aFlags[1] immediatamente prima della disallocazione frame della funzione m.

Quanti valori true sono presenti e immediatamente prima della disallocazione del frame della funzione m?

4)

Indicare il numero di volte in cui la funzione m è stata chiamata.

Quanti valori false sono presenti e immediatamente prima della disallocazione del frame della funzione m?

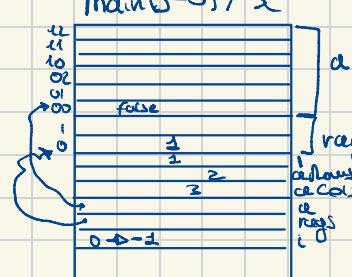
Indicare il valore di aFlags[1] immediatamente prima della disallocazione frame della funzione m.

Quanti valori true sono presenti e immediatamente prima della disallocazione del frame della funzione m?

2 + NO, 2!

main (s=0)

rags



main(j=0) / x

rags

main(j=1) / x

rags

a[2-1][1]

a[2-1][0] => a[1][0]

PROB 1 • ES. D'ESAME: ASSEGNAZIONI

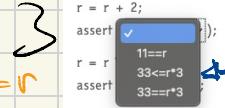
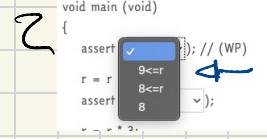
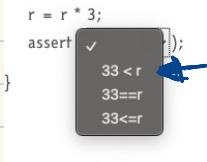
2

Dato il seguente sorgente C, usando il ragionamento backward, scegliere dai menu a tendina:

1. il predicato da asserire in ogni "assert";
2. la pre-condizione;

affinché la verità della pre-condizione implichia la verità della post-condizione.

```
void main (void)
/* pre-condizione :  */
post-condizione: 33 <= r
*/
void main (void)
{
    assert (); // (WP)
    r = r + 2;
    assert ();
    r = r * 3;
    assert ();
}
```



Post cond: $33 \leq r$ Pre $4 \leq r$
void main()

$$33 \leq (r+3)+2, 11 \leq r+2, 4 \leq r$$

$$r = r + 2;$$

$$33 \leq r+3, 11 \leq r$$

$$r = r * 3;$$

$$\leftarrow 33 \leq r$$

4

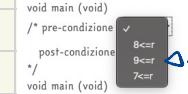
PROB 1 • ES. D'ESAME: ASSEGNAZIONI

Dato il seguente sorgente C, usando il ragionamento backward, scegliere dai menu a tendina:

1. il predicato da asserire in ogni "assert";
2. la pre-condizione;

affinché la verità della pre-condizione implichia la verità della post-condizione.

```
void main (void)
/* pre-condizione :  */
post-condizione: 33 <= r
*/
void main (void)
{
    assert (); // (WP)
    r = r + 2;
    assert ();
    r = r * 3;
    assert ();
}
```



Post = $33 \leq r$

void main 4

$$r = r + 2; \quad 33 \leq (r+3)+2 \vee 11 \leq r+2 \vee 4 \leq r$$

$$r = r * 3; \quad 33 \leq r * 3 \vee 11 \leq r$$

$$\leftarrow 33 \leq r$$

4

PROG 1 ESERCIZIO: ASSEGNAZIONI

Dato il seguente codice C, usando il ragionamento backward, scegliere dai menu a tendina:
1. il predicato da asserire in ogni "assert";
2. la pre-condizione;

affinché la verità della pre-condizione implichia la verità della post-condizione.

```
void main (void)
/* pre-condizione :  */
post-condizione: 33 <= r
*/
void main (void)
{
    assert (); // (WP)
    r = r + 2;
    assert ();
    r = r * 3;
    assert ();
}
```

void main () {

$$r = r + 2; \quad r \leq r + 2 \vee r \leq r \quad \text{OK}$$

$$r = r * 3; \quad 33 \leq r * 3 \vee r \leq r \quad \text{OK}$$

$$33 \leq r \quad \text{OK}$$

$$\text{Post} = 33 \leq r$$

Dato il seguente sorgente C, usando il ragionamento backward, scegliere dai menu a tendina:

1. il predicato da asserire in ogni "assert";
2. la pre-condizione;

affinché la verità della pre-condizione implichia la verità della post-condizione.

```
void main (void)
/* pre-condizione :  ✓
post-condizione: r == 33 */
void main (void)
{
    assert (); // (WP) ✓
    r = r + 2;
    assert (); ✓
    r = r * 3;
    assert (); ✓
}
```

main () {

$$r = r + 2; \quad r \geq r \vee r + 2 == r$$

$$r = r * 3; \quad r * 3 == 33 \vee r == r$$

$$r == 33$$

$$\text{Post} = r == 33$$