Fondamenti di Informatica T-1

Esercitazione di preparazione al compito scritto (Modulo 1)

Esempio di Prova Scritta

La prova scritta è composta da ***alcuni esercizi per un totale di 12 punti, sufficienza con 7 punti*** (durata: circa 1h).

Le tipologie di esercizi possibili possono comprendere (ma è solo un esempio):

- ***sintesi*** di una (o più) funzione ricorsiva/iterativa, che potrà contenere liste o altre strutture dati, ~~*pile o code*~~ (da accedere tramite rappresentazione interna a puntatori o tramite operazioni primitive e non primitive)
- ***analisi*** di un programma
- record di attivazione
- rappresentazione binaria
- grammatiche
- domande di teoria
- …

Contenuto di questa esercitazione

- Esercizi di sintesi
- Esercizi sulle grammatiche
- Esercizi sulla rappresentazione dei numeri
- Esercizi di analisi
- Esercizi sui record di attivazione

- Per esempi di compiti e relative soluzioni si acceda al materiale didattico del corso su **virtuale.unibo.it**

Esercizio di sintesi

Si scriva una funzione ricorsiva `crossSelection()` che, ricevute in ingresso due liste di interi positivi `l1` e `l2`, restituisca una terza lista (eventualmente non ordinata) contenente gli interi di `l2` che sono nelle posizioni indicate dai valori di `l1` (si assuma per convenzione che il primo elemento di una lista sia in posizione 1)

Ad esempio, date due liste: `l1=[1,3,4]` e `l2=[2,4,6,8,10,12]`, la lista risultante deve contenere gli elementi di `l2` che sono in prima, terza e quarta posizione, cioè: `[2,6,8]`

La lista risultante conterrà -1 qualora l'elemento in l1 non corrisponda a nessuna posizione valida. Ad esempio, date due liste: `l1=[1,3,15]` e `l2=[2,4,6,8,10,12]`, la lista risultante sarà: `[2,6,-1]`

Esercizio di sintesi

A tal scopo si realizzi una funzione ricorsiva di supporto **`select()`** che, ricevuti in ingresso una lista e un intero positivo rappresentante una posizione, restituisca l'intero della lista posto alla posizione specificata. La funzione deve restituire -1 qualora l'intero passato non corrisponda a nessuna posizione valida (si assuma comunque positivo l'intero passato)

Le funzioni **`crossSelection()`** e **`select()`** devono essere realizzate in modo ricorsivo, utilizzando il tipo di dato astratto **`list`**. Si possono utilizzare le sole operazioni primitive definite durante il corso (che quindi possono NON essere riportate nella soluzione). Non si possono usare altre funzioni di alto livello

Soluzione esercizio di sintesi

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"

/* Versione con ADT list */
int select(list l, int pos) {
      if (empty(l)) return -1;
      else if (pos == 1) return head(l);
      else return select(tail(l), pos-1);
}

list crossSelection(list l1, list l2) {
      if (empty(l1))
            return emptylist();
      else
            return cons(select(l2, head(l1)),
                  crossSelection(tail(l1), l2));
}
```

Main di prova:

```
int main()
{
 list l1,l2,l3;
l1=cons(1,cons(21,cons(15,emptylist())));
l2=cons(2,cons(4,cons(6,cons(8,cons(10,cons(12,emptylist()))))))
;
    l3=crossSelection(l1,l2);
        while(!empty(l3))
{
               printf("%d\n",head(l3));
                l3=tail(l3);
}
return 0;
}
```

Soluzione esercizio di sintesi (puntatori)

```
#include <stdio.h>
#include <stdlib.h>

typedef struct list_element {
      int value;
      struct list_element *next;
} item;
typedef item *list;
```

Soluzione esercizio di sintesi (puntatori)

```
/* Versione con puntatori */
int select(list l, int pos) {
      if (l==NULL) return -1;
      else if (pos == 1) return l->value;
      else return select(l->next, pos-1);
}

list crossSelection(list l1, list l2) {
      list l;
      if (l1==NULL) return NULL;
      else{
            l=(list) malloc(sizeof(item));
            l->value=select(l2, l1->value);
            l->next= crossSelection(l1->next, l2);
            return l;
      }
}
```

Esercizio sintesi 20 Settembre 2020

- Siano date due liste `l1` e `l2` di interi. Si assuma che le due liste abbiano pari lunghezza. Si scriva una funzione **RICORSIVA**
- `list nuovaLista (list l1, list l2);`
- che ritorni una nuova lista `l3` avente la stessa lunghezza di `l1` e `l2`, dove l'elemento k-esimo vale **il prodotto** tra il k-esimo elemento di `l1` ed il k-esimo elemento di `l2`.

Soluzione (1)

```
#include <stdlib.h>
#include "list.h"
list nuovaLista(list l1, list l2) {
    if (empty(l1) ) {
        return emptylist();   }
          else {
     return cons((head(l2) * head(l1)),
 nuovaLista(tail(l1), tail(l2)));
    }
}
```

Esercizio sintesi (cont.) 20 Settembre 2020

- Si scriva poi anche una funzione **ITERATIVA**
- `int contaNeg(list P);`
- che data una lista `P` di interi, dia in uscita un numero che rappresenta quanti elementi negativi compaiono in `P`.

Soluzione (2)

```
int contaNeg(list P){
      int n=0;
      while (!empty(P)){
            if (head(P) < 0) { n++; }
            P=tail(P);
      }
      return(n);
 }
```

Esercizio sintesi (cont.) 20 Settembre 2020

- Si realizzi una funzione `main()` che crei le liste `l1=[3,-5,2]`, `l2=[-3,1,2]`, ed utilizzi opportunamente la funzione `nuovaLista` per creare la nuova lista `l3 =[-9,-5,4]`.
- Successivamente la funzione `main()` utilizzi correttamente la funzione `contaNeg` con argomento `l3` per contare gli elementi negativi contenuti in `l3` stampando poi il risultato (2 nel nostro caso).
- Le funzioni dovranno essere implementate utilizzando le primitive dell'ADT lista, includendo `"list.h"`.

Soluzione (3)

```
int main() {
        list l3;
        list l1 = emptylist();
        list l2 = emptylist();
        l1 = cons(2,l1);
        l1 = cons(-5,l1);
        l1 = cons(3,l1);
        l2 = cons(2,l2);
        l2 = cons(1,l2);
        l2 = cons(-3,l2);
        l3 = nuovaLista(l1,l2);
        printf("%d\n", contaNeg(l3));
        return 0;
}
```

Esercizio Grammatiche (1)

Si consideri la grammatica G con scopo S, simboli non terminali {A, B, C, X, Y, Z} e simboli terminali {f, g, h, k, l, m, 2, 3, 4}:

```
S  ::=  AB  |  CA
A  ::=  YA  |  Y
B  ::=  ZC  |  YB
C  ::=  XC  |  ZY
X  ::=  k   |  l   |  m
Y  ::=  2   |  3   |  4
Z  ::=  f   |  g   |  h
```

La stringa "**342hkmg3**" appartiene al linguaggio generato da tale grammatica? In caso affermativo, se ne mostri la derivazione left-most.

SOLUZIONE:

```
S -> AB -> YAB -> 3AB -> 3YAB -> 34AB -> 34YB
-> 342B -> 342ZC -> 342hC -> 342hXC -> 342hkC
-> 342hkXC -> 342hkmC -> 342hkmZY -> 342hkmgY
-> 342hkmg3
```

Esercizio - RAPPRESENTAZIONE

Un elaboratore rappresenta i numeri interi su 8 bit in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

39 + (- 91)

Esercizio - RAPPRESENTAZIONE

+39 -> 00100111

91 -> 01011011

10100100

10100101 -> -91

Conversione del valore assoluto in base 2

Inversione dei bit

Aggiungo 1

Si esegue la somma:

```
 00100111 +
 10100101
 ------------
11001100
```

11001100

00110011

00110100 -> 52 (base 10)

Conversione in binario

```
+ 39
39: 2   |   1
19: 2   |   1
9  : 2   |   1
4  : 2   |   0
2  : 2   |   0
1  : 2   |   1
0  : 2   |   0
```

```
+ 91
91: 2   |   1
45: 2   |   1
22: 2   |   0
11: 2   |   1
5  : 2   |   1
2  : 2   |   0
1  : 2   |   1
0  : 2   |   0
```

Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione (si mostrino chiaramente)? (si motivi opportunamente in modo sintetico la risposta data)

```c
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
     int i;
     (*N)++;
     for (i=(*N)-1; i<*N; i++)
         y[i] = y[*M];
     return *N;
}
```

```c
int main () {
     char s[] = "Paperone";
     char ss[DIM]; int i=3, N = 2;
     N = (++N)-2;
     N = Funz(s, &N, &i);
     printf("N vale adesso: %d\n",N);
     {
         for (i=0; i<DIM; ++i) {
                 *(ss+i) = s[i];
                 printf("%c", *(ss+i));
         }
     }
     return 0;
}
```

Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                    *(ss+i) = s[i];
                    printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

s = | P | a | p | e | r | o | n | e | \0 |

ss= | | | | | | | | | |

i = 3

N = 2

Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

s =	P	a	p	e	r	o	n	e	\0
ss=									

i = 3

N = ~~2~~ ~~3~~ 1

Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```c
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
    int i;
    (*N)++;
    for (i=(*N)-1; i<*N; i++)
        y[i] = y[*M];
    return *N;
}
```

```c
int main () {
    char s[] = "Paperone";
    char ss[DIM]; int i=3, N = 2;
    N = (++N)-2;
    N = Funz(s, &N, &i);
    printf("N vale adesso: %d\n",N);
    {
        for (i=0; i<DIM; ++i) {
            *(ss+i) = s[i];
            printf("%c", *(ss+i));
        }
    }
    return 0;
}
```

y =
M =
N =
i =

s = | P | a | p | e | r | o | n | e | \0 |

ss= | | | | | | | | | |

i = 3

N = ~~2~~3~~1~~

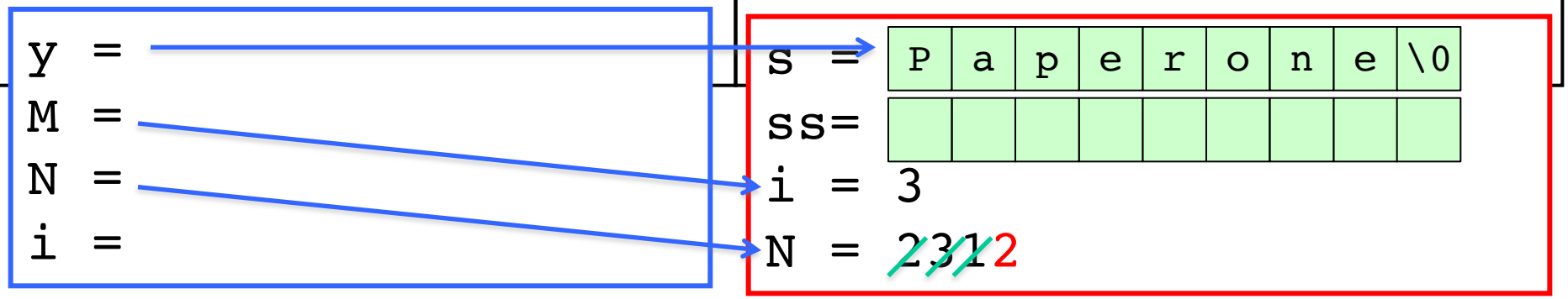
Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```c
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```c
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```



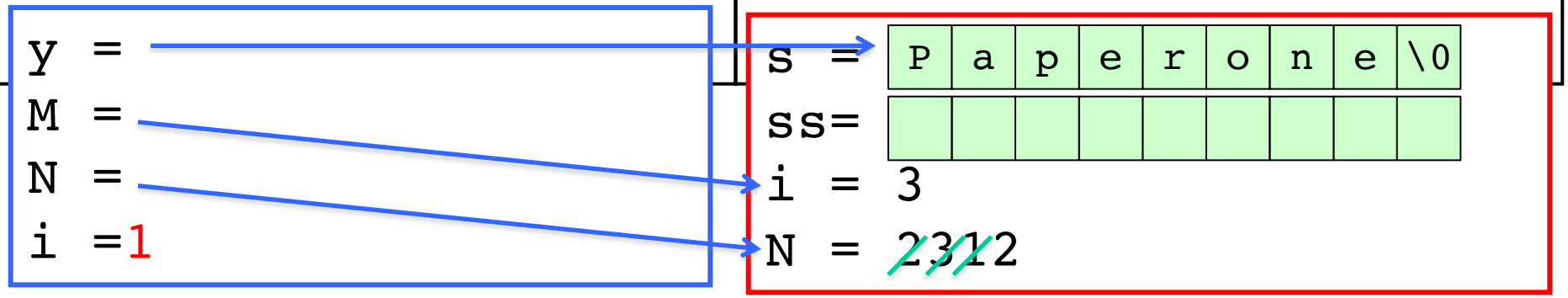
Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```



Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```c
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
         y[i] = y[*M];
      return *N;
}
```

```c
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

y =
M =
N =
i =1

s = P a p e r o n e \0
ss=
i = 3
N = 2312

Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

y =
M =
N =
i = 1 ~~2~~

s = P ~~a~~ e p e r o n e \0
ss=
i = 3
N = ~~2 3~~ 1 2

Esercizio analisi (7)

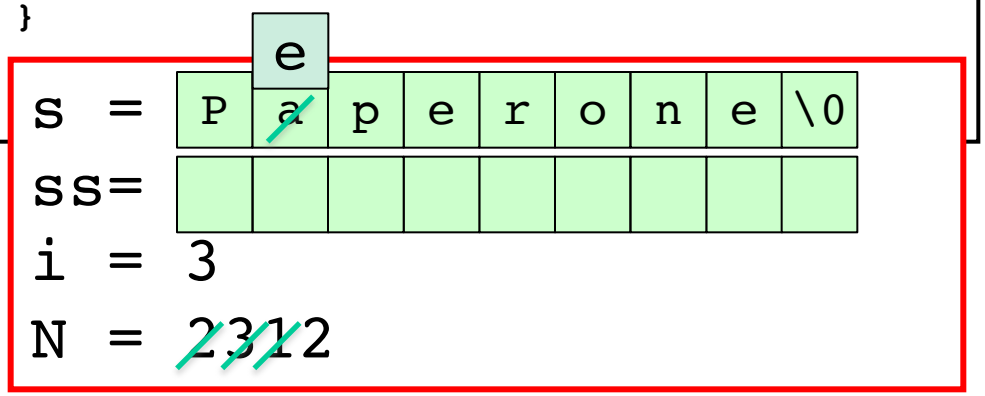
Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

```
N vale adesso: 2
```



Esercizio analisi (7)

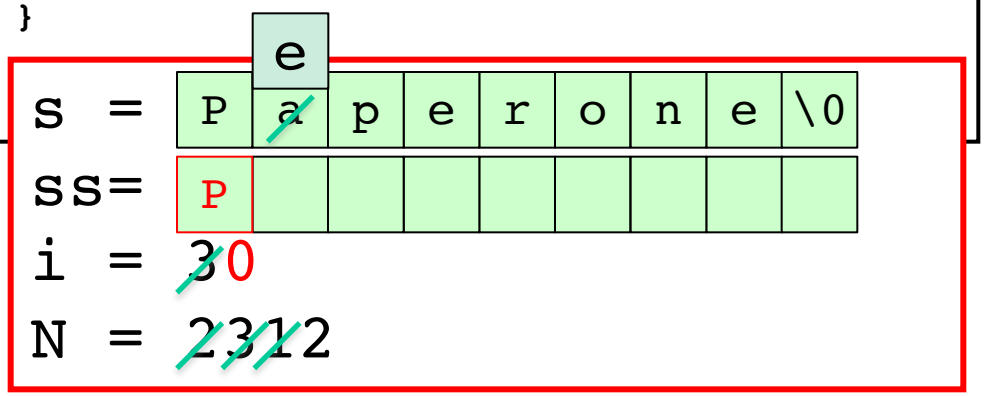
Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

```
N vale adesso: 2
```



Esercizio analisi (7)

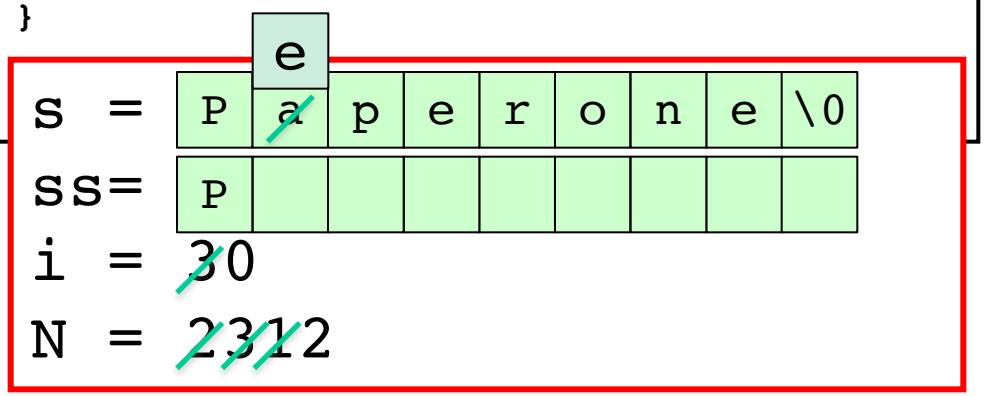
Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

```
N vale adesso: 2
P
```



Esercizio analisi (7)

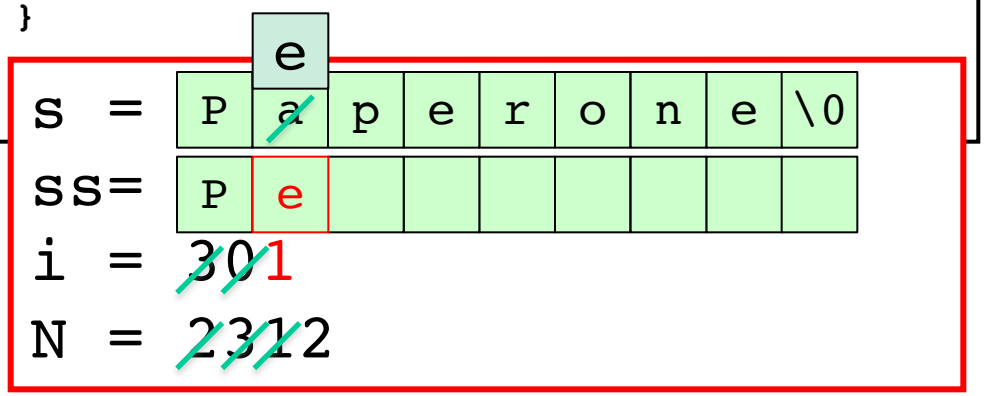
Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

```
N vale adesso: 2
Pe
```



Esercizio analisi (7)

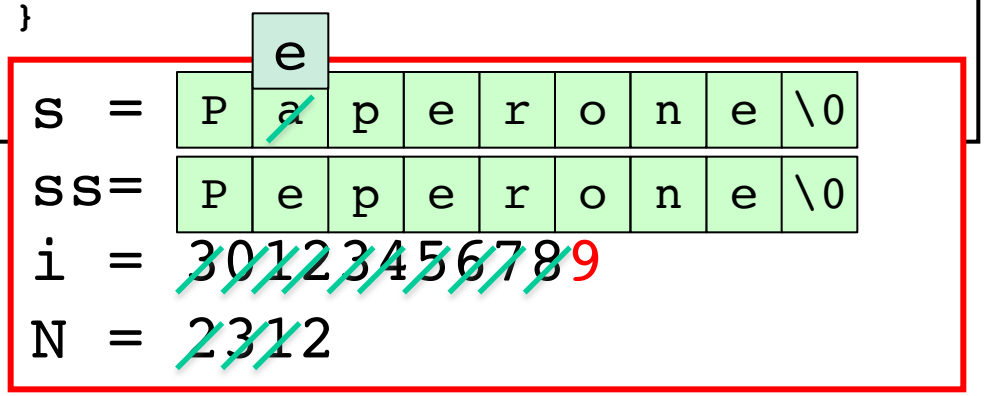
Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
      int i;
      (*N)++;
      for (i=(*N)-1; i<*N; i++)
          y[i] = y[*M];
      return *N;
}
```

```
int main () {
      char s[] = "Paperone";
      char ss[DIM]; int i=3, N = 2;
      N = (++N)-2;
      N = Funz(s, &N, &i);
      printf("N vale adesso: %d\n",N);
      {
          for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
          }
      }
      return 0;
}
```

```
N vale adesso: 2
Peperone\0
```



Esercizio analisi (7)
Soluzione

Il programma compila correttamente e stampa:

```
N vale adesso: 2
Peperone
```

- La funzione **`Funz`** incrementa il valore riferito da **`N`**, e poi assegna all'elemento in posizione **`*N-1`** del vettore **`y`** il valore in posizione **`*M`**; restituisce infine il nuovo valore riferito da **`N`**
- Il **`main`** inizia dichiarando alcune vairabili, instanziando **`N`** a **`2`** e poi modificandolo con un assegnamento al valore **`1`**
- Viene poi invocata la funzione **`Funz`**, con parametri (**`"Paperone",1,3`**); per quanto detto, viene modificato l'elemento in posizione **`1`** (**`'a'`**), a cui viene assegnato il valore in posizione 3 (**`'e'`**)
- Inifine l'array **`s`** viene copiato (elemento per elemento) nell'array **`ss`**, che viene poi stampato a video

Esercizio Record di Attivazione (8)

Si consideri la seguente funzione:

```
int funzione(float a, float b) {
    a = a+b;
    b--;
    if (b > 0)
       return funzione(a, b);
    else
       return a;
}
```

Si scriva il risultato della funzione quando invocata come:

```
funzione(2.5, 3.5)
```

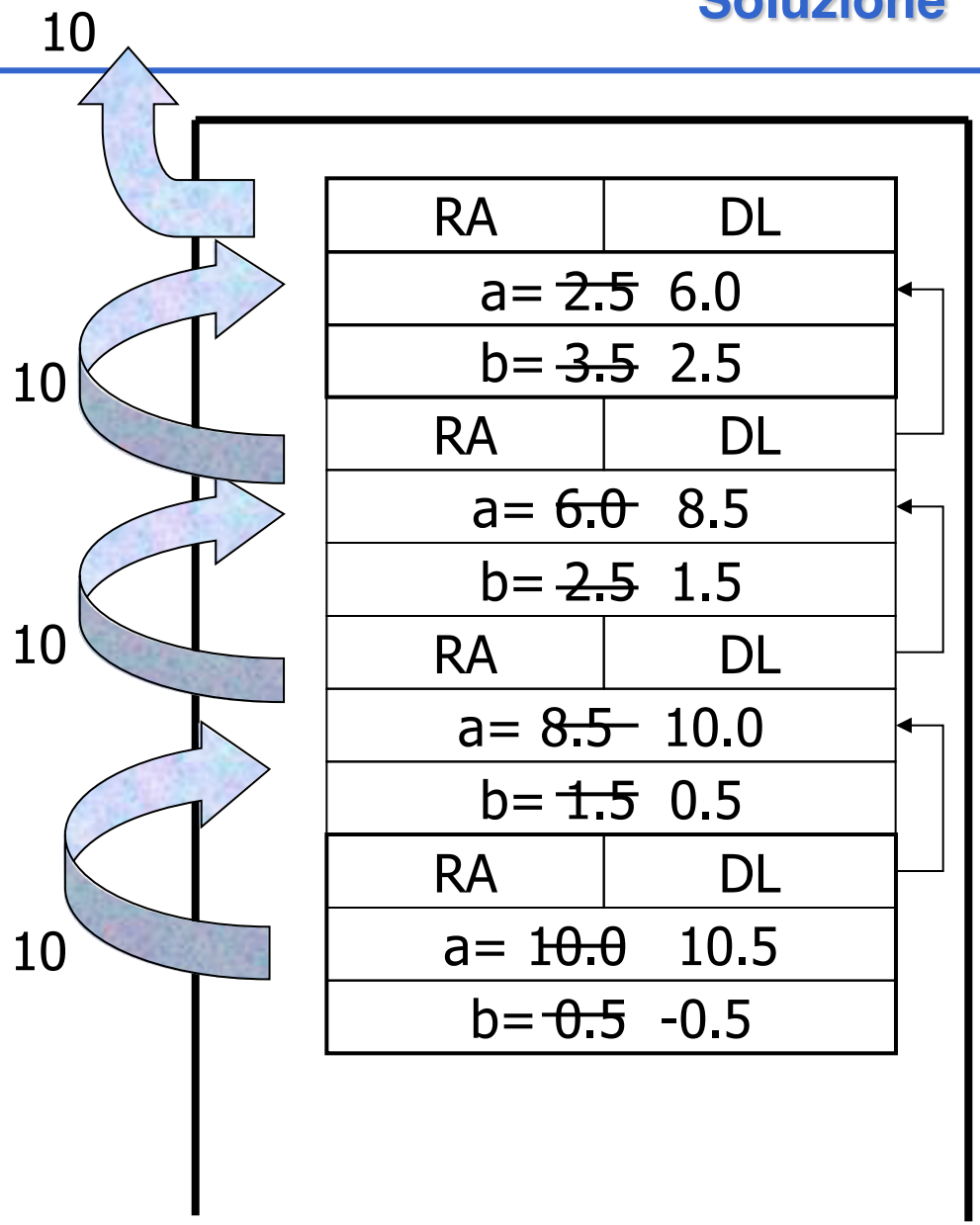
e si disegnino i corrispondenti record di attivazione (si faccia particolare attenzione al fatto che la funzione restituisce un valore intero).

Esercizio Record di Attivazione (8)

```
int funzione(float a, float b) {
    a = a+b;
    b--;
    if (b > 0)
       return funzione(a, b);
    else
       return a;
}
```

funzione(2.5, 3.5) -> funzione(6, 2.5)->
funzione(8.5,1.5) -> funzione(10, 0.5) =
10,5 troncato a 10

Esercizio Record di Attivazione (8)
Soluzione




```
int funzione(float a,
float b) {
  a = a+b;
  b--;
  if (b > 0)
        return
        funzione(a, b);
  else
        return a;
}
```

```
Invocazione  funzione(2.5,
3.5)
funzione(2.5, 3.5) ->
funzione(6, 2.5)->
funzione(8.5,1.5) ->
funzione(10, 0.5) =
```

10,5 troncato a 10

Esercizio Record di Attivazione (9)

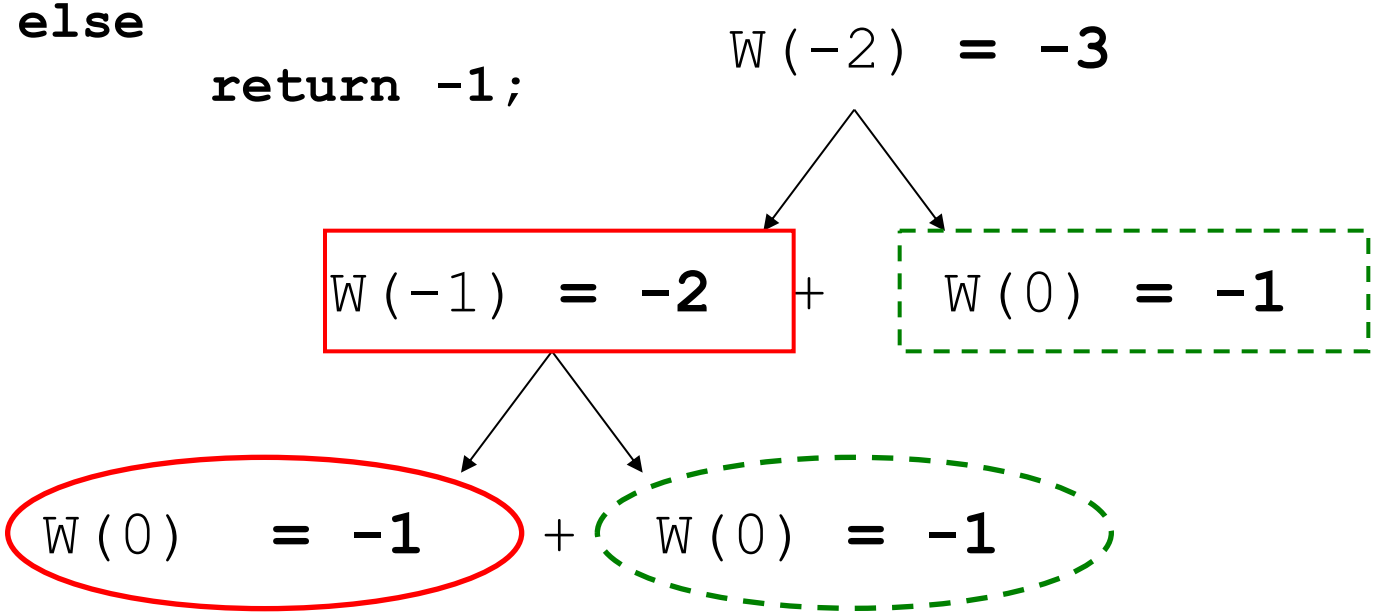
Si consideri la seguente funzione **W**:

```
double W(int x){
   if (x<0) {
          x++;
          return W(x)+W(x/2);
          x++;
   }
   else
          return -1;
}
```

Si scriva il risultato della funzione quando invocata come **W(-2)** e si disegnino i corrispondenti record di attivazione.

Soluzione record di attivazione

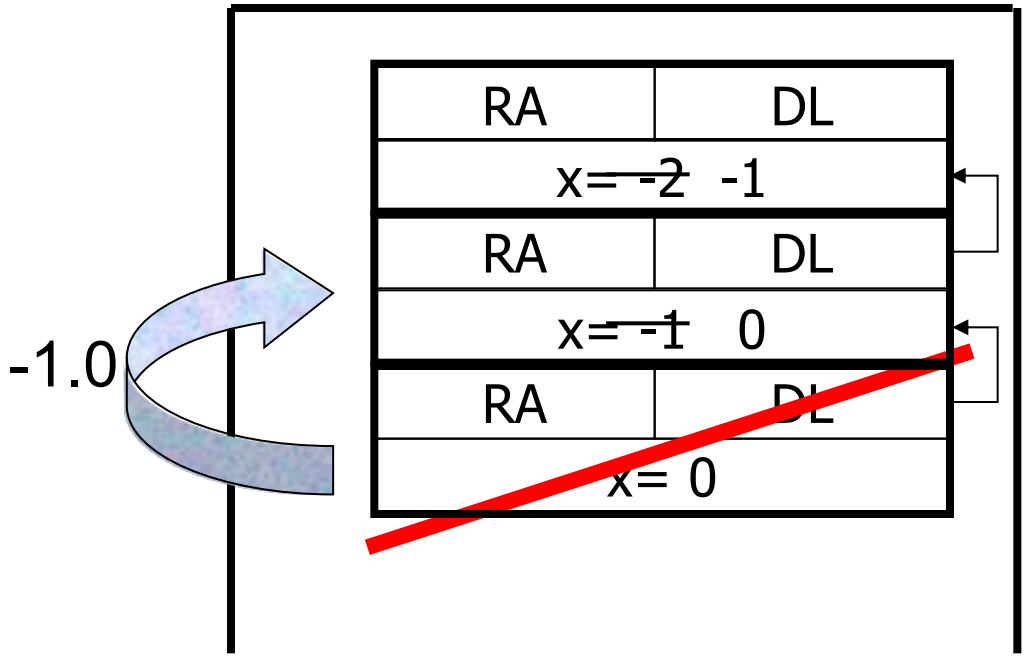
```
double W(int x){
      if (x<0) {
            x++;
            return W(x)+W(x/2);
            x++;
      }
      else
            return -1;
  }
```



Nota il secondo x++ non viene mai eseguito

Esercizio Record di Attivazione (9)
Soluzione

Supponendo che la valutazione degli addendi nella somma venga fatta a partire da sinistra, si ottiene prima:



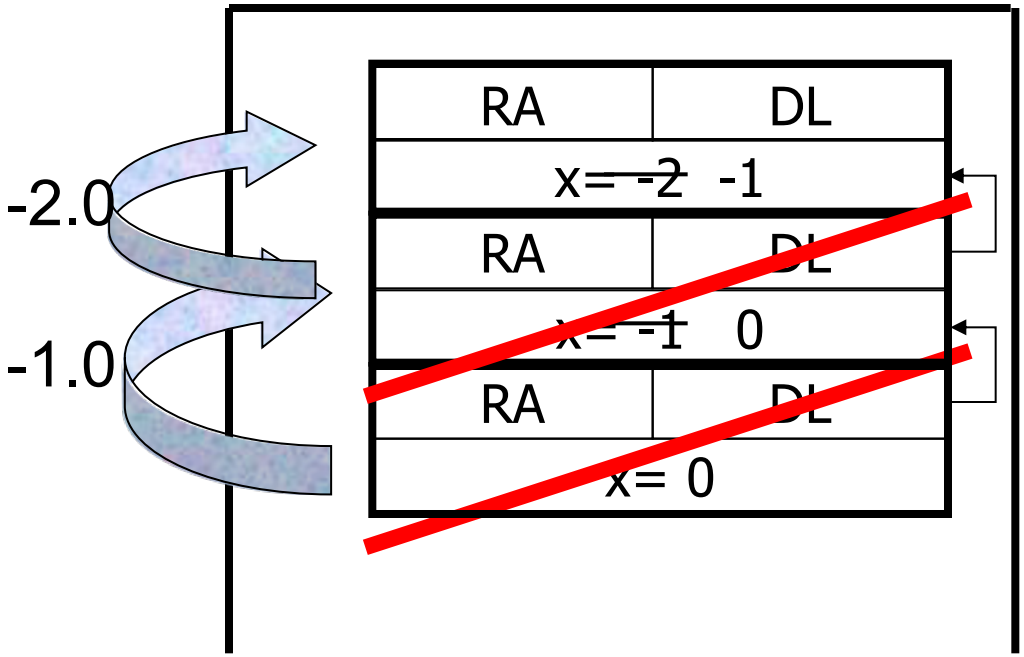

```
double W(int x){
  if (x<0) {
    x++;
    return W(x)+W(x/2);
    x++;
  }
  else
    return -1;
}
```

prima di qua

poi l'eliminazione dell'ultimo record, e …

Esercizio Record di Attivazione (9)
Soluzione

Supponendo che la valutazione degli addendi nella somma venga fatta a partire da sinistra, si ottiene prima:



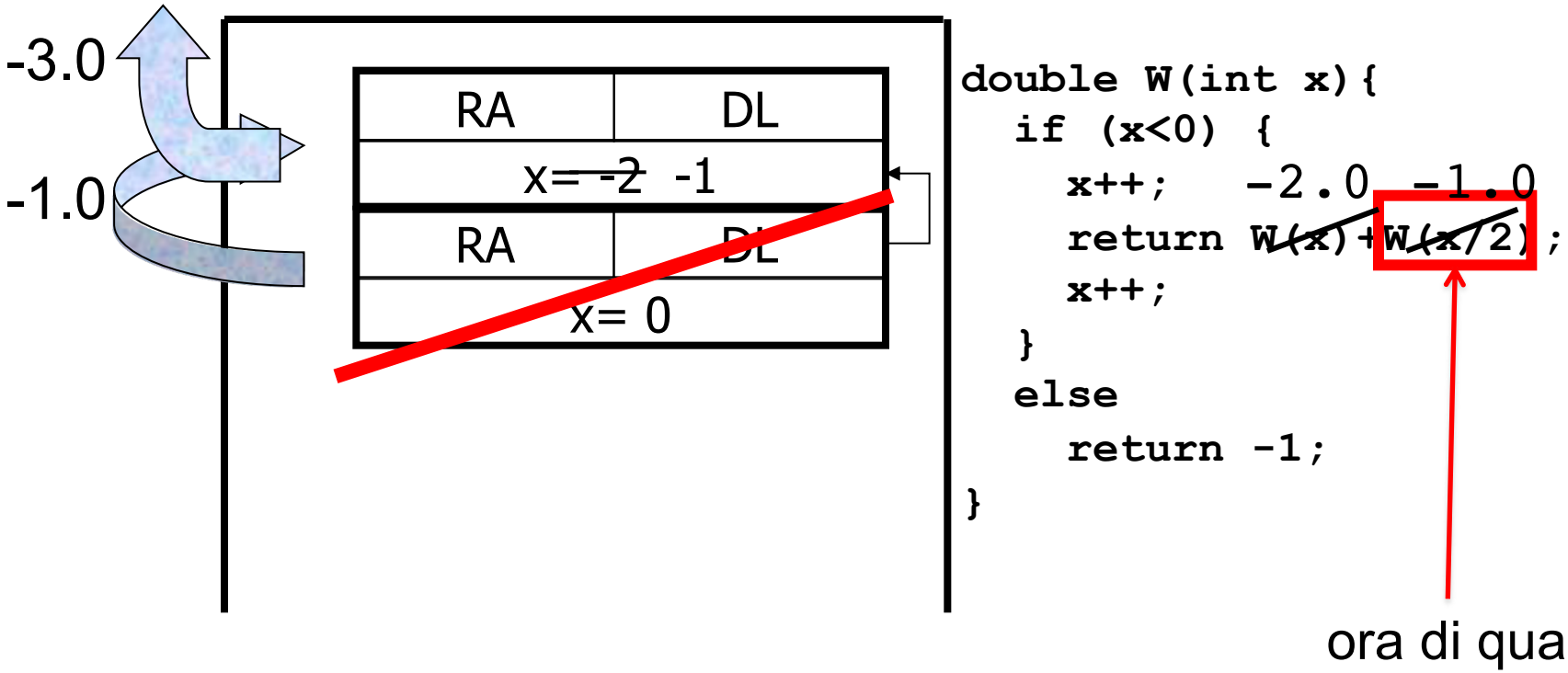

```
double W(int x){
  if (x<0) {
    x++;    -1.0  -1.0
    return W(x)+W(x/2);
    x++;
  }
  else
    return -1;
}
```

ora di qua

Quindi viene fatta la prima somma con risultato -2.00, restituita indietro

Esercizio Record di Attivazione (9)
Soluzione

…nel primo record il primo addendo vale -2. Ora dobbiamo calcolare il secondo addendo: `W((int)-1.0/2)` => `W(0)`



La somma finale (**-3.0**) è restituita alla prima invocazione di W.

Alcune possibili domande di teoria

- Relative a tutto il programma del corso. Si consultino i testi degli esami precedenti su virtuale.unibo.it.

- Ad esempio su:

  - Compilatori-interpreti
  - Passaggio dei parametri
  - Linguaggi di alto-livello
  - Files
  - Allocazione dinamica …