

Esempio di Prova Scritta

La prova scritta è composta da **alcuni esercizi per un totale di 10 punti** (durata: circa 1h15'). Le tipologie di esercizi possibili comprendono:

- **sintesi** di una funzione ricorsiva/iterativa, che potrà contenere liste, 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

1

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]**

2

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

3

Soluzione esercizio di sintesi

```
/* Versione con primitive */
element 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));
}
```

4

Soluzione esercizio di sintesi (puntatori)

```
/* Versione con puntatori */
element 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;
    }
}
```

5

Esercizio di analisi

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 16
void ribalta(char a[], char *b, int *dim) {
    int i, size = 0;
    *dim = 0;
    for (size=0; a[size] != '\0'; size++);
    for (i=size-1; i>=0; i--) {
        *(b+size-i-1) = a[i];
        (*dim)++; }
    *(b+size) = '\0';
    (*dim)++;
}
```

6

Esercizio di analisi

```
int main () {
    char mese[] = "Aprile";
    char *other;
    int *value, i;

    other = (char *) malloc(DIM * sizeof(char));
    value = (int *) malloc(sizeof(int));
    *value = 0;

    ribalta(mese, other, value);

    for (i=0; i < (*value) - 1; i++)
        printf("%c", other[i]);
    printf("%d\n", *value);
    return 0;
}
```

7

Soluzione esercizio di analisi

Il programma è corretto sintatticamente, viene compilato, ed in esecuzione stampa:

```
elirpA 7
```

Nella fase iniziale del programma `main` vengono dichiarate alcune variabili e allocata memoria dinamicamente; in questa fase si inizializza anche a 0 la variabile riferita dal puntatore `value`. Quindi viene invocata la funzione `ribalta(...)`: tale funzione esegue inizialmente un ciclo con lo scopo di determinare la lunghezza della stringa `a`, e poi copia (ribaltando) il contenuto di `a` in `b`. L'operazione di inversione dell'ordine dei caratteri è effettuata tramite opportune somme/sottrazioni negli indici. La funzione termina copiando anche il terminatore di stringa (al termine della stringa), e restituisce tramite il parametro `dim` il numero di caratteri copiati (compreso il terminatore), cioè 7.

Il `main` stampa tutti i caratteri della stringa `other`, terminatore escluso, e di seguito il valore puntato da `value` che, per quanto detto, è 7

Funzione ricorsiva e record di attivazione

Data la funzione:

```
int func(int a, float b){
    int x; float y;
    if( a - b < -22 ) return a - b;
    else{
        x = a / b;
        y = a * b;
        return func(x,y) + func(x/2,y*2);
    }
}
```

e la funzione chiamante:

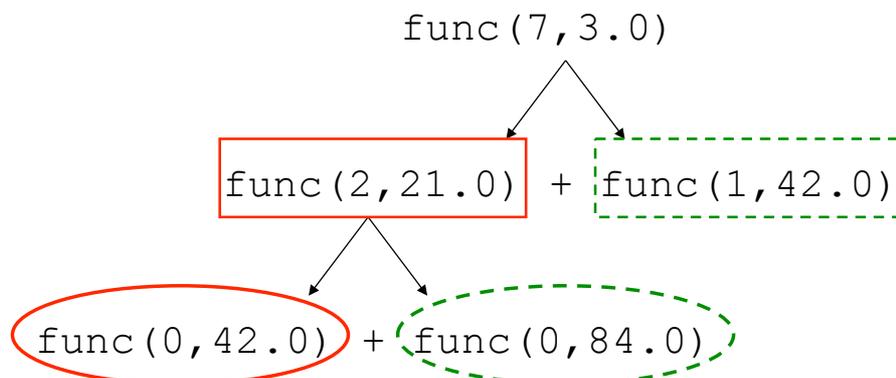
```
int main(){
    printf("%d\n", func(7.7,3.0));
    return 0; }
```

mostrare la sequenza dei record di attivazione.
Che cosa viene stampato sullo standard output?

9

Soluzione record di attivazione

La chiamata alla funzione ricorsiva non lineare `func(...)` può essere rappresentata graficamente nel seguente modo

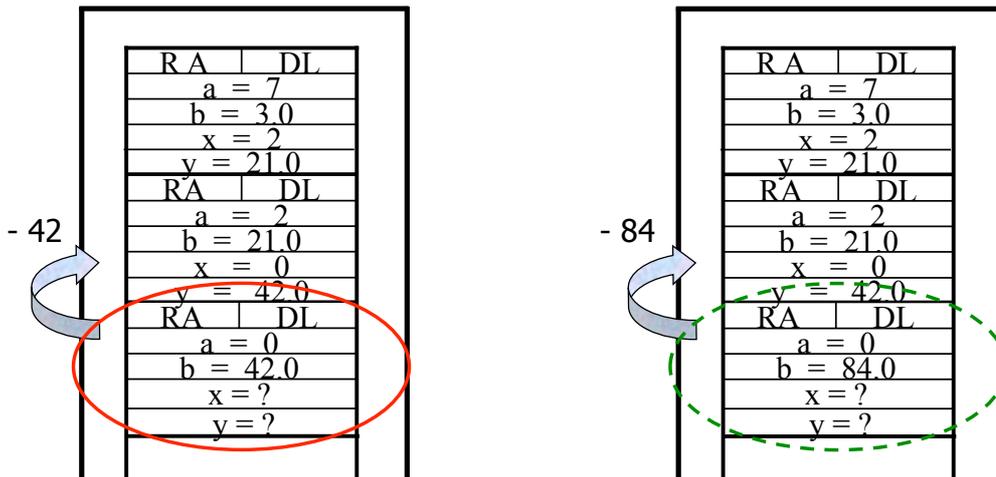


10

Soluzione record di attivazione

Nel caso in cui l'ordine di valutazione degli addendi sia da sinistra a destra:

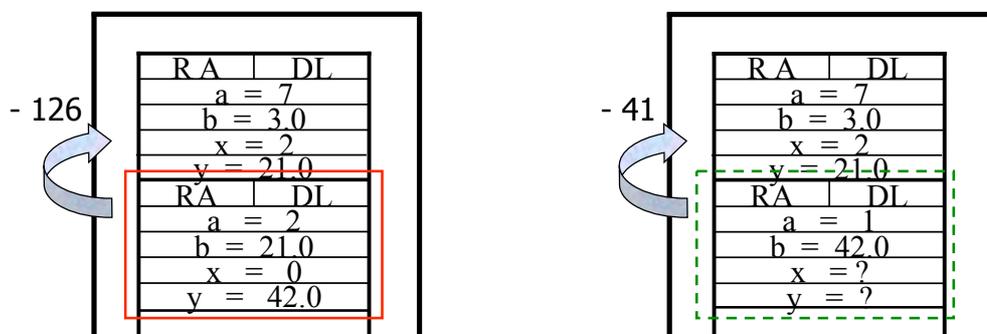
$$\text{func}(0, 42.0) + \text{func}(0, 84.0)$$



11

Soluzione record di attivazione

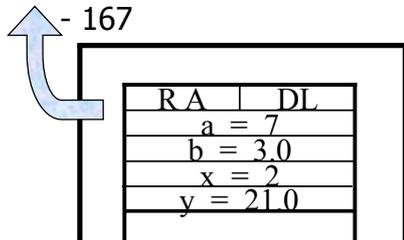
$$\text{func}(2, 21.0) + \text{func}(1, 42.0)$$



12

Soluzione record di attivazione

`func(7, 3.0)`



Sullo standard output viene scritto il risultato della chiamata alla funzione `func(...)`, ovvero **"-167"**

Che cosa sarebbe accaduto se l'ordine di valutazione degli addendi fosse stato invertito, ovvero da destra a sinistra? Il risultato sarebbe stato differente?

13

Esercizio di sintesi 2

Si scriva una funzione iterativa:

```
int fun(char *str1, char *str2)
```

che, ricevuti come parametri in ingresso due stringhe ben formate `str1` e `str2`, restituisca come valore di ritorno un `int` rappresentante la somma totale di occorrenze di ogni carattere di `str1` in `str2`. Ad esempio, la chiamata

```
fun("Pippo", "Poporono")
```

dovrà restituire 7 (1 occorrenza di 'P', 1 di 'p' per 2 volte, 4 di 'o').

Si proponga una possibile funzione chiamante

14

Esercizio di sintesi 2

```
int fun(char* str1, char* str2){
    int totale=0, i=0, j;
    while(str1[i]!='\0'){
        j=0;
        while(str2[j]!='\0'){
            if(str1[i]==str2[j]) totale++;
            j++;
        }
        i++;
    }
    return totale;}

int main(){
    int occorrenze=0;
    char str1[]="abcdefghilmno";
    char str2[]="aabbbbde";
    occorrenze=fun(str1, str2);
    printf("Occorrenze %d\n", occorrenze);
    return 0;}
```

15

Esercizio Grammatiche

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 Esercizio Grammatiche

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

Soluzione Esercizio Grammatiche

- Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 2.
- Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):

$$118 + (-43)$$

Soluzione Esercizio Grammatiche

118 ->	01110110	01110110 + (118)
+43 ->	00101011	11010101 = (-43)
	11010100	-----
-43 ->	11010101	01001011
(+75)		

Soluzione Esercizio Grammatiche

- Compilatori-interpreti
- Passaggio dei parametri
- Algoritmi di ordinamento
- Linguaggi di alto-livello
- Etc. etc. ...