Fondamenti dell'informatica
(prova in itinere)

23 aprile 2018

Esercizio 1

Abbiamo creato un nuovo linguaggio di programmazione N, molto espressivo ed elegante. Abbiamo a disposizione una macchina x86 e il compilatore $\mathrm{GCC}^{\mathrm{x86}}_{\mathrm{C,x86}}$. Il nostro obiettivo è ottenere un compilatore nativo efficiente che produca codice efficiente per il linguaggio N, $\overline{\mathrm{CN}}^{\mathrm{x86}}_{\mathrm{N,x86}}$. Siccome N è molto espressivo ed elegante, usiamo N stesso per scrivere $\mathrm{CN}^{\mathrm{N}}_{\mathrm{N,x86}}$. Poi abbiamo due alternative:

1. scriviamo un compilatore "quick and dirty" $\mathrm{QADC}^{\mathrm{C}}_{\mathrm{N,C}}$;
2. scriviamo un interprete "quick and dirty" $\mathrm{QADI}^{\mathrm{C}}_{\mathrm{N}}$.

Si completi il percorso per ciascuna alternativa.

Esercizio 2

Si consideri uno pseudolinguaggio con gestione delle eccezioni, scope statico e passaggio per valore. Si dica, motivando la risposta, cosa stampa il seguente programma.

```
int x = 1;
void g(int z) {
  x = z+1;
  throw E();
  x = z+1;
}
void foo(int y) {
  try { g(x+y); } catch (E) { write(x); }
}
void main() {
  { int x = 10;
    x = x+1;
    try { foo(x); } catch (E) { write(x); }
    try { g(x); } catch (E) { write (x); }
    try { foo(x); } catch (E) { write(x); }
    try { g(x); } catch (E) { write(x); }
  }
  write(x);
}
```

La risposta deve iniziare con un elenco di numeri interi separati da virgole e terminato da un punto, ovvvero della forma "3, 19, 2, 4. [Segue adeguata motivazione]"

Esercizio 3

Si dica, motivando adeguatamente, cosa stampa il seguente programma in uno pseudolinguaggio con passaggio per valore nei tre casi: scope statico, scope dinamico con *deep binding*, scope dinamico con *shallow binding*:

```
int u = 42;
int v = 69;
int w = 17;
proc add(z : int)
  u := v + u + z
proc bar(fun : int->void)
  int u := w;
  fun(v)
proc foo(x : int, w : int)
  int v := x;
  bar(add)
main
  foo(u, 13);
  write(u)
end;
```

La risposta deve avere la forma "sc.st.: n, sc.dyn.dp.bnd.: m, sc.dyn.sh.bnd.: p. [Segue adeguata motivazione]"

Esercizio 4

Si consideri il seguente programma, in un linguaggio non meglio identificato:

```
function phil(int a, int b, int c)
  begin
    b := b + 5;
    b := a + c + 4;
    print a, b, c;
  end

function main
  begin
    int j := 10;
    ink k := 15;
    phil(j, j, j + k);
    print j, k;
  end
```

Si dica cosa stampa il programma in ciascuna delle seguenti ipotesi:

1. tutti i parametri sono passati per valore;
2. `a` e `b` sono passati per riferimento, `c` per valore;
3. `a` e `b` sono passati per valore-risultato con copia da sinistra a destra, `c` per valore;
4. tutti i parametri sono passati per nome.

Per rispondere, innanzitutto compilare una tabella della forma

1	a_1	b_1	c_1	d_1	e_1
2	a_2	b_2	c_2	d_2	e_2
3	a_3	b_3	c_3	d_3	e_3
4	a_4	b_4	c_4	d_4	e_4

e, poi, motivarne adeguatamente il contenuto.

Esercizio 5

Si considerino le seguenti definizioni in C:

```
struct S {
  int x;
  char y;
  short z;
  int w[4];
};
struct T {
  char a[4];
  struct S b[4];
};
struct T A[4];
```

Si assuma `sizeof(char) == alignof(char) == 1`, `sizeof(short) == alignof(short) == 2`, `sizeof(int) == alignof(int) == 4`. Si scriva un espressione il cui valore sia l'indirizzo di `A[i].b[j].w[k]` assumendo che l'indirizzo iniziale di `A` sia 0. Si scrivano poi gli indirizzi dei seguenti lvalue sotto le stesse ipotesi:

1. `A[1].b[2].w[3]`
2. `A[1].b[3].w[2]`
3. `A[2].b[1].w[3]`
4. `A[3].b[2].w[1]`

Esercizio 6

Si considerino gli operatori **div** e **rem** in un linguaggio di programmazione non meglio specificato. Assumiamo di sapere che **div** e **rem** soddisfano le seguenti condizioni: per ogni $D, d \in \mathbb{Z}$, se $q = D \mathbf{\ div\ } d$ e $r = D \mathbf{\ rem\ } d$, allora

$$q, r \in \mathbb{Z}, \tag{1}$$

$$D = d \cdot q + r, \tag{2}$$

$$|r| < |d|. \tag{3}$$

Queste informazioni sono sufficienti a definire la semantica di **div** e **rem** come funzioni? Se sì, formalizzarla; se no, definire quante più possibili semantiche per **div** e **rem** che siano plausibili e rispettino le condizioni (1)–(3). Cosa succede se sostituiamo la condizione (3) con la seguente?

$$0 \leq r < |d|. \tag{4}$$

Esercizio 7

Si spieghi cosa si intende per "ricorsione di coda". Si scriva una versione ricorsiva di coda per ciascuna delle seguenti funzioni in un linguaggio funzionale non meglio specificato:

```
fun fact(n) = if n=0 then 1
              else n*fact(n-1)
fun exp(x, n) = if n=0 then 1
                else x*exp(x, n-1)
fun fib(n) = if n=1 then 1
             else if n=2 then 1
             else fib(n-1) + fib(n-2)
fun real(n) = if n=0 then 0.0
              else 1.0 + real(n-1)
```