Corso di Linguaggi di Programmazione
Prova scritta del 28 Maggio 2024.

Tempo a disposizione: ore 2.
Svolgere gli esercizi 1–4, 5–8 su due fogli separati.
Scrivere nome, cognome e matricola su ogni foglio consegnato.

Foglio **1** ▷ 1. Si consideri il seguente frammento in uno pseudolinguaggio con parametri di ordine superiore:

```
{int x = 1;
 int h(){
    write(x);
    }
 void foo (int f(), int n){
    int x = 3;
    int g(){
    write(x);
    }
    if (n==0)   {f();
       g();
     }
    else      {x = 4;
    foo(g,0);
    }
    }
 {int x = 2;
  foo(h,1);
  }
}
```

Si dica cosa stampa il frammento con con scope statico e deep binding.

Foglio **1** ▷ 2. Si consideri il seguente frammento di programma scritto in uno pseudo-linguaggio che usa scope statico implementato mediante catena statica

```
{
void f() {
    void g() {
    corpo_di_g;
    }

 void h() {
     void l(){
     corpo_di_l;
     }
   corpo_di_h;
    }
corpo_di_f;
} }
```

Si descriva graficamente l'evoluzione della catena statica nella sequenza di chiamate f, h, l, g, h supponendo che tutte le chiamate rimangano attive (ossia nessuna funzione ha restituito il controllo).

Foglio **1** ▷ 3. Si assuma che in un generico linguaggio imperativo a blocchi, che ammette solo ricorsione in coda, la funzione f contenga una chiamata della funzione g. Il numero dei record di attivazione (RdA) presenti a run-time sulla pila fra il RdA di f e quello della chiamata di g è fissato staticamente o può variare dinamicamente? Motivare la risposta.

Foglio **1** ▷ 4. Si consideri un linguaggio con scope statico, implementato mediante display, nel quale gli identificatori sono noti staticamente. Si prendano in considerazione le operazioni di: (1) "accesso ad una variabile non locale x" e (2) "accesso ad una variabile locale y" (nel contesto di un blocco). Per ognuna delle due operazioni si dica, motivando brevemente la risposta, se ognuna delle seguenti affermazioni è vera o falsa: i tempo necessario all'esecuzione dell'operazione è proporzionale al

(a) numero di variabili presenti nel programma;

(b) numero di variabili presenti nei blocchi compresi tra quello di dichiarazione della variabile e quello in cui si accede alla variabile stessa;

(c) valore dell'indice, nel display, del blocco che contiene la variabile a cui si accede;

(d) numero di blocchi che contengono testualmente il blocco in cui si accede alla variabile e che sono contenuti in quello nel quale la variabile è dichiarata;

(e) il tempo è costante, e quindi indipendente da questi parametri.

Foglio **2** ▷ 5. Viene dato un linguaggio con tipaggio nominale implicito (indicato dal simbolo `?`) e polimorfismo di sottotipo e parametrico. Sono dati i tipi `A`, `B` e `C` entrambi sottotipi di `A` (`B <: A` e `C <: A`) e `List[ T ]` che supporta le operazioni `add: T -> ()` e `get: () -> T`, con `T` parametro di tipo. Indicare quali istruzioni nel codice di `f` sono corrette e scorrette per il controllore di tipi. Per l'inferenza dei tipi impliciti dei parametri di `f` si usi l'algoritmo di unificazione, usando le equazioni riportate a sinistra di `f` dove `X`, `Y` e `Z` corrispondono ai parametri corrispondenti (in minuscolo) di `f`. Spiegare in breve il ragionamento seguito.

```
X -> X -> Y = C -> K
Z -> K = H
H = B -> C -> Y
Y -> B -> Z = A -> B -> B
```

```
f( ? x, ? y, ? z, List[A] aa, List[B] bb  ){
 y = x;               /*I1*/
 y = bb.get();        /*I2*/
 x = z;               /*I3*/
 aa.add( bb.get() ); /*I4*/
 aa = bb;             /*I5*/
 aa.add( x );         /*I6*/ }
```

Foglio **2** ▷ 6. Usando uno pseudolinguaggio che usi puntatori, si fornisca un esempio di memory leak. Si spieghi, tramite l'esempio, se la tecnica delle "lock and keys" può ovviare a questa classe di problemi.

Foglio **2** ▷ 7. Viene dato uno pseudolinguaggio dove **`new`** crea un nuovo oggetto nello heap. La struttura `Array` occupa 2 byte, da sommare allo spazio di ogni elemento contenuto: un `A` occupa 2 byte e un `B` occupa 4 byte. `Array` offre `add`, che aggiunge un elemento. Il linguaggio usa stop-and-copy garbage collection, attivando la collection superata la soglia del 50% di occupazione della memoria disponibile al collector. Lo heap ha 28 byte di memoria complessiva (inizialmente vuota). Indicare, spiegando brevemente il ragionamento seguito, quante volte viene chiamato il garbage collector nell'esecuzione del seguente frammento di codice.

```
Array bb = new Array();
for( int j = 0; j < 3; j++ ) {
 Array aa = new Array();
 for ( int i = j; i < 3; i++ ) {
  aa.add( new A() );
 }
 bb.add( new B() );
}
```

Foglio **2** ▷ 8. Si rappresenti e descriva la struttura delle vtable corrispondenti alle seguenti dichiarazioni di classi con ereditarietà singola (**`class`** `B` **`extends`** `A` indica ereditarità dalla classe `A` alla classe `B`) e classi astratte.

```
abstract class A {
 int x = 2;
 int a(){ return x; }
 abstract int b( int x );
 int c(){ return a(); };
}
```

```
class B extends A {
 int x = 3;
 int c(){
  return b( this.x ) / x;
 }
 int b( int x ){
  return b( super.x );
 }
}
```

```
class C extends B {
 int b( int x ){
  return a()*(x + super.x);
 }
}
```

Inoltre, indicare, spiegando il ragionamento, il valore stampato dalle istruzioni `A a = new C(); print( a.c() );`