Corso di Linguaggi
Prova scritta parziale del 25 Gennaio 2021.

Tempo a disposizione: ore 2.

1. Nel seguente frammento di codice si indichi la modalità di passaggio dei parametri, si inseriscano al posto simboli &, £ dei parametri attuali e si inserisca al posto di $ un qualsiasi elemento (che possa stare a destra di un assegnamento) per far si che il programma stampi 2, 4 e 8.

```
int i = 3;
int[] A = new int[ 5];
A[0] = 2;
or (j = 1; j <= 4; j+= 1)
        {A[j] = A[j-1]*2};

void fie (int x, int  y) {
         int i = 2;
         write(y);
          i = $;
          write(y);
          i = $;
          write(y);
        }

{int i = 0;
fie ( &, £);

}
write (A[i] )
```

2. Si consideri il seguente frammento in uno pseudolinguaggio con parametri di ordine superiore:

```
{int x = 10;
 int h(){
    write(x);
    }
 void foo (int f(), int n){
    int x = 30;
    int g(){
    write(x);
    }
    if (n==0)   {f();
                 g();
                }
    else        {x = 40;
                foo(g,0);
                }
    }
 {int x = 20;
  foo(h,1);
  }
}
```

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

3. Si consideri il seguente frammento di programma scritto in uno pseudo-linguaggio che usa scope statico.

```
{
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 del display nella sequenza di chiamate f, h, l, g, h supponendo che tutte le chiamate rimangano attive (ossia nessuna funzione ha restituito il controllo).

4. Si dica, motivando la risposta, cosa viene stampato dall'esecuzione del `main` della seguente classe `Test` in Java.

```
class Y extends Throwable {
   int x=20;
}
class X extends Y {
   int x=20;
}
 class C {
  void f() throws X, Y {
   throw new X();
   }
void g (int sw) throws X, Y {
   if (sw == 0) {f();
     throw new X(); }
   try {f();} catch (Y e) {System.out.println("in␣g");}
   }
}

public class Test {
        public static void main(String[] args) throws X, Y {
        C c = new C();
        try {c.g(1);}
        catch (X e)  {System.out.println("in_main");}
         }
}
```

5. Si considerino le seguenti dichiarazioni (Pascal):

```
type stringa = packed array [1..16] of char;
type punt_stringa = ^stringa;
type persona = record
        nome = stringa;
        case studente: Boolean of
            true:  (matricola: integer);
            false: (codicefiscale: punt_stringa)
end;
```

   e si supponga che la variabile C contiene il puntatore alla stringa "CODICEJOHN". Si descriva il layout di memoria dopo ognuna delle seguenti istruzioni e si dica se: i) il compilatore segnala qualche errore di tipo e ii) si verifica un qualche errore din qualche punto dell'esecuzione (delle seguenti istruzioni)

```
...
var john persona;

john.studente:= true;
john.matricola := 123123;
john.studente:= true;
john.codicefiscale := C;
```

6. Si fornisca un esempio di oggetto denotabile la cui vita sia piú lunga di quella dei legami (nomi, puntatori, ecc.) che vi si riferiscono.

7. Usando uno pseudolinguaggio che ammetta l'uso dei puntatori si fornisca un frammento di codice che generi un "dangling reference". Si faccia quindi vedere come con la tecnica dei "locks and keys" non si ha più tale problema. Avremmo potuto usare la tecnica del "contatore dei riferimenti" invece del "locks and keys" ? Motivare la risposta.

8. Si discuta brevemente la differenza esistente fra polimorfismo universale parametrico e polimorfismo universale di sottotipo.