Quarta esercitazione
Linguaggi di programmazione

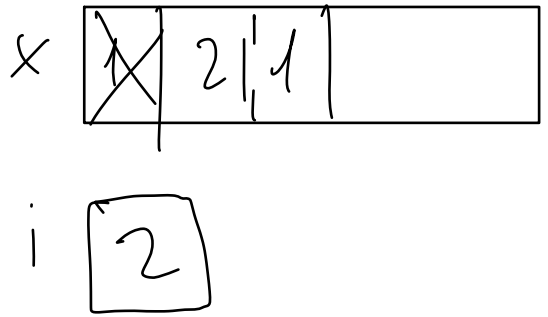
Tutor didattico: Giosuè Cotugno
giosue.cotugno2@unibo.it
A.A. 2023/2024

Esercizio 1

Cosa stampa (indicando i passaggi del ragionamento seguito) il seguente frammento di codice scritto in un linguaggio che ammette parametri per riferimento. Nel codice, `T*` è il tipo "puntatore a un valore di tipo `T`", `T[]` è il tipo "array di elementi di tipo `T`", `print` è una funzione che stampa il valore di qualsiasi sequenza di parametri dati in input, separati da virgola e `&var` e `*var` corrispondono rispettivamente a referenziamento e dereferenziamento di una variable `var`.

```
void foo ( int* x, int* y, int* j ){
  x[ *j ] = *j + 1;
  print( *y );// 2
  x++;
  x[ *j ] = *j;
  print( x[ *j ], *j ); // 1, 1
  *j = *j + 1;
}

int[] x[ 10 ];
int i = 1;
x[ 0 ] = 1;
x[ 1 ] = 2;
x[ 2 ] = 3;
foo( x, &x[ i ], &i );
print( x[ i ], i ); // 1, 2
```



Esercizio 2

LISTA<T>

SL:T

LISTA<A> C: LISTA<TOP>

Si consideri il seguente frammento di codice, in un sistema di tipi nominale, dove `A` e `B` sono tipi non confrontabili e valgono le relazioni di sottotipaggio `A <: Top` e `B <: Top` e la notazione `X[? <: T]` e `X[? :> T]` indicano rispettivamente polimorfismo parametrico vincolato covariante e controvariante rispetto al tipo proprio `T`.

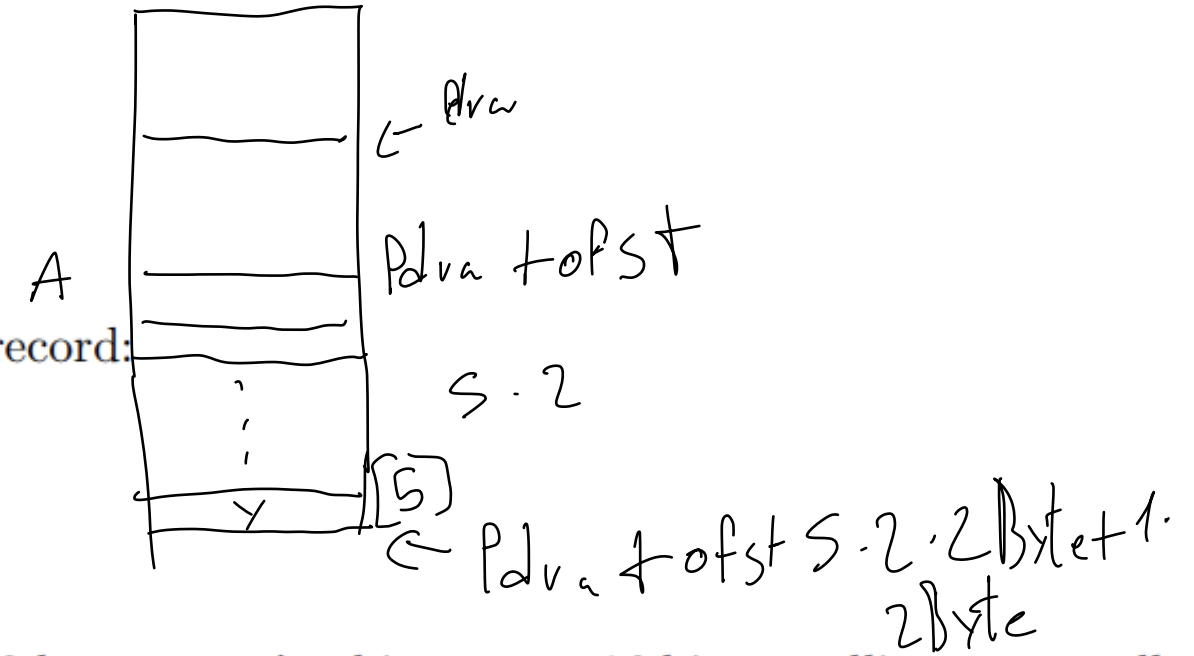
```
f( List< A > a, List< B > b ){
 List< B > b = a;          // I1 X
 List< Top > at = a;       // I2 X
 List< A > a1 = b;         // I3 X
 List< ? :> A > a2 = a;    // I4 ✓
 List< ? <: B > b1 = a1;   // I5 X
}
```

A : TOP

LISTA<A> > : LISTA<TOP>

Si dica quali istruzioni (`I1`, ...) e perchè verrebbero dichiarate non corrette da un type checker.

Esercizio 3



6. Si consideri la seguente definizione di tipo record:

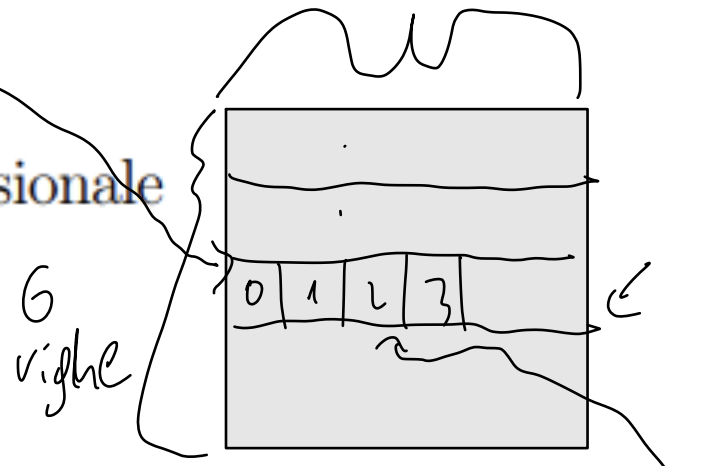
```
type S = struct{
    int x;
    int y;
    };
```

Si supponga che un `int` sia memorizzato su 2 byte, su un'architettura a 16 bit con allineamento alla parola. In un blocco viene dichiarato un vettore:

```
S A[10];
```

Indicando con `PRDA` il puntatore all'RdA di tale blocco, e con `ofst` l'offset tra il valore di `PRDA` e l'indirizzo iniziale di memorizzazione di `A`, si dia l'espressione per il calcolo dell'indirizzo dell'elemento `A[5].y` (indicare tutte le costanti in decimale).

Esercizio 4



Si consideri la seguente dichiarazione di array multidimensionale

```
int x;
read ( x );
int A [ 10 ][ x ];
```

dove il comando `read(x)` permette di leggere il valore della variabile `x` dall'esterno.

Sappiamo che: un intero è memorizzato su 4 byte; l'array è memorizzato in ordine di riga (raw major), con indirizzi di memoria crescenti (cioè se un elemento è all'indirizzo `i`, il successivo è a `i + 4`, ecc.); il valore di `x` letto è 6.

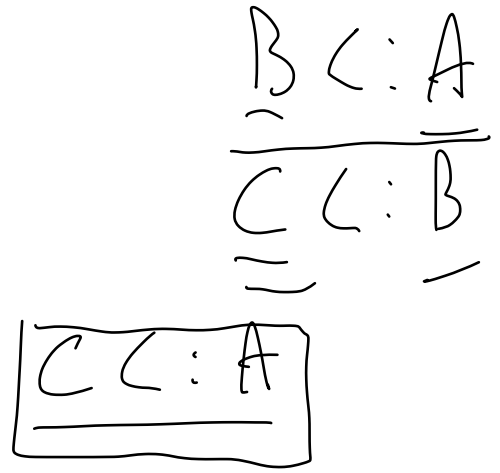
Come viene memorizzato l'array `A`? Qual'è l'offset dell'elemento `A[2][3]` rispetto all'inizio dell'array? (Si risponda in notazione decimale).

Esercizio 5

8. Si assuma un linguaggi di programmazione a oggetti, con tipi nominali e passaggio per riferimento. Le classi `A`, `B`, e `C` sono tali che `B` è sottoclasse di `A` e `C` è sottoclasse di `B`. Nel linguaggio, il tipo `T[]` indica un array di oggetti della classe `T` con scritture e letture covarianti rispetto ai sottotipi.

   Indicare quali istruzioni verrebbero segnate come *non* corrette dal controllore dei tipi (e indicare brevemente perché).

```
void f( A a, B b, C c, A[] aa, B[] bb, C[] cc ) {
 c = b;              // I1 ✗
 a = b;              // I2 ✓
 c = bb[ 0 ];        // I3 ✗
 aa = bb;            // I4 ✓
 a = aa;             // I5 ✗
 bb = bb;            // I6 ✓
 a = bb[ 0 ];        // I7 ✓
 aa[ 0 ] = cc;       // I8 ✗
 b = bb;             // I9 ✗
 aa[ 0 ] = cc[ 0 ];  // I10 ✓
 cc = bb;            // I11 ✗
 c = bb[ 0 ];        // I12 ✗
}
```



Esercizio 6

5. In un linguaggio con passaggio per riferimento e supporto al polimorfismo di sottotipo e parametrico, sono dati i seguenti tipi per cui vale la relazione di sottotipaggio `<:` nelle seguenti direzioni: `Mammal <: Animal`, `Lion <: Carnivore <: Mammal` e `Giraffe <: Herbivore <: Mammal`. Viene inoltre definito il contenitore polimorfo `Cage[ T ]` dotato delle operazioni `add: T -> ()` e `remove: () -> T`, con `T` parametro di tipo. Il linguaggio offre l'istruzione `new` per creare una nuova istanza di un tipo e supporta sottotipi parametrici con la notazione `T[? <: S]` e `T[? :> S]` per indicare la relazione di sottotipaggio del tipo parametrico `?` rispetto ad un tipo concreto `S`.

   Nel codice sottostante, indicare quali istruzioni sono errate e spiegare brevemente perchè.

```
Cage[ ? :> Carnivore ] cage1 = new Cage[ Mammal ]();    // I1
Cage[ ? <: Mammal ]    cage2 = new Cage[ Carnivore ](); // I2
Cage[ ? <: Herbivore ] cage3 = new Cage[ Giraffe ]();   // I3
cage1.add( new Lion() );                                // I4
Mammal a1 = cage2.remove();                             // I5
cage1.add( new Giraffe() );                             // I6
cage2.add( cage1.remove() );                            // I7
cage2.add( a1 );                                        // I8
Herbivore a2 = cage3.remove();                          // I9
cage2 = cage3;                                          // I10
cage3.add( a2 );                                        // I11
cage1.add( cage3.remove() );                            // I12
```