German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail

CSEN 1003 Compiler, Spring Term 2020
Practice Assignment 10

Exercise 10-1

Determine the types and relative addresses for the identifiers in the following sequence of declarations:

```
float x;
record ( float x; float y; ) p;
record ( int tag; float x; float y; ) q;
```

Solution:

Identifier	**Relative Address**	**Type**
x	0	$float$
p.x	8	$float$
p.y	16	$float$
p	8	$record([\langle x, 0, float\rangle; \langle y, 8, float\rangle])$
q.tag	24	$integer$
q.x	28	$float$
q.y	36	$float$
q	24	$record([\langle tag, 0, integer\rangle; \langle x, 4, float\rangle; \langle y, 12, float\rangle])$

Exercise 10-2

Assuming that function *widen* in lecture 9 slide 25 can handle any of the types in the hierarchy (a) in lecture 9 slide 23, translate the expressions below. Assume that c and d are characters, s and t are short integers, i and j are integers, and x is a float.

a) `x = s + c.`

b) `i = s + c.`

c) `x = (s + c) * (t + d).`

Solution:

We assume the type of the right-side expression of an assignment to be coerced into the type of the left-side identifier.

a) `x = s + c`

   Solution.

```
t1 = (int) s
t2 = (int) c
t3 = t1 + t2
t4 = (float) t3
x = t4
```

b) `i = s + c`

   Solution.

```
t1 = (int) s
t2 = (int) c
t3 = t1 + t2
i = t3
```

c) `x = (s + c) * (t + d)`

   Solution.

```
t1 = (int) s
t2 = (int) c
t3 = t1 + t2
t4 = (int) t
t5 = (int) d
t6 = t4 + t5
t7 = t3 * t6
t8 = (float) t7
x = t8
```

Exercise 10-3

Consider the following polymorphic function:

```
fun reverse(x) =
    if length(x)==1 then x
    else append(head(x), reverse(tail(x))
```

What is the type of `reverse`? Show the type inferences and the unifications computed by the type inference algorithm in Lecture 9 Slide 30.

Solution:

Expression	Type	Unification
`reverse`	$\alpha_1 \to \beta_1$	
`x`	α_1	
`if`	$bool \times \alpha_2 \times \alpha_2 \to \alpha_2$	
`length`	$list(\alpha_3) \to int$	
`length(x)`	int	$\alpha_1 = list(\alpha_3)$
`==`	$\alpha_4 \times \alpha_4 \to bool$	
`length(x) == 1`	$bool$	$\alpha_4 = int$
`x`	$list(\alpha_3)$	$\alpha_2 = list(\alpha_3)$
`append`	$\alpha_5 \times list(\alpha_5) \to list(\alpha_5)$	
`append(head(x), reverse (tail(x)))`	$\alpha_2 = list(\alpha_3)$	$\alpha_5 = \alpha_3$
`head`	$list(\alpha_6) \to \alpha_6$	
`head(x)`	$\alpha_5 = \alpha_3$	$\alpha_6 = \alpha_5 = \alpha_3$
`reverse(tail(x))`	β_1	$\beta_1 = list(\alpha_5) = list(\alpha_3)$
`tail`	$list(\alpha_7) \to list(\alpha_7)$	
`tail(x)`	$list(\alpha_3)$	$\alpha_7 = \alpha_3$

Therefore, `reverse` has the type $list(\alpha_3) \to list(\alpha_3)$.