Sample Solution for Homework 9

Problem 1 Parameter Passing Modes (12 Points)

This problem is meant as a warm-up exercise for Problem 2 where you will implement a version of JakartaScript that supports different parameter passing modes.

Consider the typed variant of JakartaScript that supports the different parameter passing modes we discussed. For each of the following programs, say whether the program is well-typed. If not, provide a brief explanation of the type error. If yes, compute the value that the program evaluates to and provide a brief explanation why you obtain that specific value.

(a)

```
1 const y = 3;
2 const f = function(let x: number) {
3     x = x + 1; return x + y;
4   };
5 f(y) + y
```

This program is well-typed and evaluates to `10`. The function `f` is called with the value `3`, which is then stored in a **`let`** parameter. The assignment on line 3 only updates that **`let`** parameter to `4`. Thus, the call to `f` returns `7`. Hence, the program evaluates to `7 + 3 = 10`.

(b)

```
1 let x = 3;
2 const f = function(name x: number) {
3     return x + x;
4   };
5 f(x = x + 1) + x
```

This program is well-typed and evaluates to `14`. Since the function `f` takes a by-name parameter. The assignment expression `x = x + 1` is evaluated twice in the call to `f`. The call therefore returns `9` and the new value of `x` after the call is `5`. Hence, the program evaluates to `9 + 5 = 14`.

(c)

```
1 let y = 3;
2 const f = function(let x: number) {
3     x = x + 1; return x + y;
4   };
5 f(y * 2) + y
```

This program is well-typed and evaluates to `13`. The function `f` is called with the value `6`, which is stored in a **`let`** parameter. This variable is then incremented to `7` by the

assignment on line 3. The value of `y` is left unmodified. The function thus returns `7+3=10` and the program evaluates to `10+3=13`.

(d)

```
1 let x = 3;
2 const f = function(ref y: number) {
3     y = y + 1; return y + x;
4   };
5 f(x) + x
```

This program is well-typed and evaluates to `12`. The function `f` is called with the mutable variable `x` as argument. Since `f` takes a **`ref`** parameter, the assignment on line 3 sets the value of both `x` and `y` to `4`. Thus, the call to `f` returns `8`, and the program evaluates to `8 + 4 = 12`.

(e)

```
1 let y = 3;
2 const f = function(ref x: number) {
3     x = x + 1; return x + y;
4   };
5 f(y * 2) + y
```

This program is not well-typed. The function `f` takes a call-by-reference parameter and must therefore be called with a mutable variable as argument. However, it is called with the expression `y * 2`.