Exercises for week 2

1. Draw the derivation tree for each of the following programs if it exists (you are allowed to compress subtrees whose border word is a single letter or digit). Otherwise argue why no derivation tree for the $C0$ grammar exists.

   (a)
   ```
   int c;
   int i;
   int main()
   {
   int j;
   if c<i {i = 4} else {i = 0; j = c};
   return d
   }
   ```
   (20 credit points)

   (b)
   ```
   int main() {return 4};
   int function(int x) {return 9*x};
   ```
   (10 credit points)

   (c)
   ```
   int b;
   int main() {return -1};
   b = main()
   ```
   (10 credit points)

2. A $C0$ program starts with

   ```
   typedef uint[64] a;
   typedef uint[16] b;
   typedef struct {a gpr; b spr} pcb;
   typedef pcb[q+1] PCB_0;
   ```

   Specify the corresponding portion of the type table tt. In general you can present type tables as a table or write type 'equations'. What components of $\$gm$ do you know already? (15 credit points)

3. Specify the function table for the function f, which is given below. Do not forget $tt(\$f)$.

   ```
   int f(int z)
   {
   ```

```
int result;
if z<18433 {result = 95}

else {if z>18400 {result = -95}};

return result
}
```

z	$ft(z).t$	$ft(z).p$	$ft(z).VN$	$ft(z).body$	$tt(\$z)$
f					

(15 credit points)

4. Consider the following program (numbers of lines are comments and not part of the program).

```
0: typedef int* intp;
1: typedef intp[10] intparr;
2: typedef intparr* intparrp;
3: intparr a;
4: int fak(int x)
5: {
6:   int y;
7:   if x==1 {y = x} else
8:   {
9:     y = fak(x-1);
10:     y = x*y
11:   };
12:   return y
13: };
14: int main()
15: {
16:   intparrp b;
17:   int i;
18:   b = a&;
19:   i = 1;
20:   while i<11
21:   {
22:     b*[i-1]* = fak(i);
23:     i = i + 1
24:   };
25:   return 26
26: }
```

(a) Draw the derivation tree for the type declaration in line 1.
(10 credit points)

(b) Draw the derivation tree for the assignment in line 22. (10 credit points)

(c) Specify the type table and function table. (20 credit points)

5. Recall derivation trees that encode sequences, i. e., derivation trees using production rules $\langle XS \rangle \rightarrow X \mid X \circ \langle XS \rangle$, where X is a nonterminal and $\circ$ is some terminal. Let $u \in \mathbb{N}^*$ be the root of such a derivation tree.

   (a) Recall, for $i \in \mathbb{N} \setminus \{0\}$, the definition of $se(u, i)$ that gives the ith sequence element of the tree starting in u. Consider the following (alternative) recursive definition:

$$\begin{aligned} se(u, 1) &= u \circ 0, \\ se(u, i + 1) &= se(u \circ 2, i). \end{aligned}$$

   Prove that $se(u, i) = u \circ 2^{i-1} \circ 0$. (10 credit points)

   (b) We define $fseq(u) = se(u, 1) \circ \ldots \circ se(u, n)$. This turns such a derivation tree into the list of its sequence elements. Give a formal definition of $fseq(u)$, i.e., a definition that does not use three dots. (10 credit points)