Assignment 6 - Sander & Joran

Opgave 18

ex. ABC

```
factorial:                  ;        ; function header
    esr 1                   ;      2 ; 1 local variable
    iload 0                 ;      2 ; load x
    iloadc_1                ;      1 ; load 1
    ile                     ;      1 ; x <= 1
    branch_f else           ;      3 ; if (x <= 1) {
    iloadc_1                ;      1 ;       load 1
    istore 1                ;      2 ;       res = 1
    jump end                ;      3 ; }
else:                       ;        ; else {
    iload 0                 ;      2 ;       load x
    isrg                    ;      1 ;       factorial(_)
    iload 0                 ;      2 ;             load x
    iloadc_1                ;      1 ;                  load 1
    isub                    ;      1 ;                  x-1
    jsr 1 factorial         ;      4 ;            factorial(x-1)
    imul                    ;      1 ;       x * factorial(x-1)
    istore 1                ;      2 ;       res = x * factorial(x-1)
end:                        ;             ; }
    iload 1                 ;      2 ; load res
    ireturn                 ;      1 ; return res
```

ex. D

```
                ;       ; function header
esr 1           ;   0+2 ; 1 local variable
iload 0         ;   2+2 ; load x
iloadc_1        ;   4+1 ; load 1
ile             ;   5+1 ; x <= 1
branch_f 8      ;   6+3 ; if (x <= 1) {
iloadc_1        ;   9+1 ;   load 1
istore 1        ;  10+2 ;   res = 1
jump 16         ;  12+3 ; }
                ;       ; else {
iload 0         ;  14+2 ;   load x
isrg            ;  16+1 ;   factorial(_)
iload 0         ;  17+2 ;     load x
iloadc_1        ;  19+1 ;      load 1
isub            ;  20+1 ;      x-1
jsr 1 -21       ;  21+4 ;    factorial(x-1)
imul            ;  25+1 ;   x * factorial(x-1)
istore 1        ;  26+2 ;   res = x * factorial(x-1)
                ;       ; }
iload 1         ;  28+2 ; load res
ireturn         ;  30+1 ; return res
```

Opgave 19

```
C[
   for(int Id = Expr1,Expr2){
     Body
   }
   Rest
 ][n]
 =>             //*n is the string representation of n
 int _i_*n;
 int _upper_*n;
 int _inc_*n;
 _i_*n = Expr1;
 _upper_*n = Expr2;
 _inc_*n = 1;
 while((_inc_*n > 0 && _i_*n<_upper_*n)||(_inc_*n < 0 && _i_*n > _upper_*n)){
   C[Body][n+1]
   _i_*n = _i_*n + _inc_*n;
 }
 C[Rest][n+k+1] //where k is the amount of forloops in C[Body][n+1]
```

```
C[
   for(int Id = Expr1,Expr2,Expr3){
     Body
   }
   Rest
 ][n]
 =>             //*n is the string representation of n
 int _i_*n;
 int _upper_*n;
 int _inc_*n;
 _i_*n = Expr1;
 _upper_*n = Expr2;
 _inc_*n = Expr3;
 while((_inc_*n > 0 && _i_*n<_upper_*n)||(_inc_*n < 0 && _i_*n > _upper_*n)){
   C[Body][n+1]
   _i_*n = _i_*n + _inc_*n;
 }
 C[Rest][n+k+1] //where k is the amount of forloops in C[Body][n+1]
```