1.

E : Exp -> Store -> (Store X $\mathbb{Z}$) $\perp$

E [Num]s = (s,N[Num])

E [Var]s = $\perp$ if s(Var)= $\perp$

=(s, s(Var)) otherwise

E [E1 op E2]s = $\perp$ if *E* [E1]s = $\perp$ or *E* [E2]s = $\perp$

=(s2, a O[O] b) where *E* [E1]s =(s1,a) and *E* [E2]s1 =(s2,b) otherwise

E [Var=Exp]s = $\perp$ if *E* [Exp]s = $\perp$

=(s2,v) where *E* [Exp]s=(s1,v) and *s2(Var)=v* and *s2(I) = s1(I) for I* $\neq$ *Va*r

otherwise

C : Com -> Store -> Store $\perp$

C [Exp]s = $\perp$ if *E* [Exp]s = $\perp$

=t where *E* [Exp]s =(t,v) otherwise

C [while Exp do Com]s

= $\perp$ if *E* [Exp]s = $\perp$

=t if *E* [Exp]s =(t,0)

= $\perp$ if *E* [Exp]s =(t,v) and v $\neq$ 0 and *C*[Com]t= $\perp$

=*C* [while Exp do Com](*C* [Com]t) if *E* [Exp]s =(t,v) otherwise

2.

```
fun palindrome x =
    let fun reverse x =
        let rev(0,x) = x | rev(x,y) = rev (x div 10, y*10+x mod 10);
        in rev(x,0) end;
    in if x = reverse x then true else false end;
```

3.

```
fun split [] = ([],[])| split [x] = ([x],[])
    | split (x::y::z) =
        let val (l,r) = split(z)
        in (x::l,y::r) end;
fun combine x [] = x
    | combine [] x = x
    | combine (x::xs) (y::ys) =
    if(x<=y) then x::(combine xs (y::ys)) else y::(combine (x::xs) ys);
```

```
fun msort [x] = [x]
      | msort x =
            let val (l,r) = split x
            in combine (msort(l)) (msort(r)) end;
```

4.

```
fun self x = x;
fun n2c 0 f:int->int = self | n2c n f = f o (n2c (n-1) f);
fun c2n f = f (fn x=>x+1) 0;

infixr 7 **;
fun f ** g = f o g;
infixr 6 ++;
fun plus f g x s = f x (g x s);
fun f ++ g = plus f g;
```

5.

```
datatype 'a stack = Empty | Push of 'a*'a stack;
fun loop [x] = Push(x,Empty) | loop (x::xs) = Push(x,loop(xs));
fun top (Push(x,y)) = x;
fun tal (Push(x,y)) = y;
fun isdigit x = if ord (x) > 47 then
                        if ord (x) < 58 then true else false
                else false;

fun eva [] (Push(x,a)) = x
| eva (x::xs) a = if (isdigit(x)) then eva xs (Push((ord (x) - 48),a))
                  else if x = #"+" then eva xs (Push((top(tal(a)) + top(a)), (tal(tal(a)))))
                  else if x = #"-" then eva xs (Push((top(tal(a)) - top(a)), (tal(tal(a)))))
                  else if x = #"*" then eva xs (Push((top(tal(a)) * top(a)), (tal(tal(a)))))
                  else if x = #"/" then eva xs (Push((top(tal(a)) div top(a)), (tal(tal(a)))))
                  else eva xs (Push((top(tal(a)) mod top(a)), (tal(tal(a)))));
fun eval s = eva (expolde s) Empty;
```