1)

```
proc Q(x)
integer x;
begin
integer  y := x * 2;
Q := y
end;
```

2)

We can consider w = A(i) so when P called , in the first line we have :

i := w = A(1) = 2

w := i = 2 ⇒ A(2) = 2

⇒ A(1) = A(2) = i = 2

3)

a) fun f(x,y) = if(y=0) then x else if(x=0) then y else x+y;

b) No, since no variable can occur twice in each pattern . Error: duplicate variable in pattern(s)

c) We have to assign different names for different variables and use "if clause" to check equal variables.
fun eq(x,y) = if x=y then true else false

d) since ML has type variables and as we know "*ML* is *statically* typed", meaning that it performs its type checking operations at compile time .also, functions' behavior cannot be processed at compile time, thus repeated variables are not allowed in ML.

4)

a)
```
fun maptree (f,leaf(x)) = leaf(f(x))
  | maptree (f,node(x,y)) = node(maptree(f,x),maptree(f,y));
```

b) ('a → 'b) * 'a tree → 'b tree

6)

a) fun curry f = fn x => fn y => f(x,y);

fun uncurry g = fn (x,y) => g x y;

b)

uncurry(curry(f)) = uncurry(’a → (’b →’c)) = (’a*’b) →’c
curry(uncurry(g)) = curry((’a*’b) →’c) = ’a → (’b →’c)

8)

a)

```
fun merge cmp (nil, yss) = yss

  | merge cmp (xss, nil) = xss

  | merge cmp (x::xs, y::ys) = if cmp(x,y) then x :: merge cmp (xs, y::ys)
else y :: merge cmp (x::xs, ys);
```

```
b)
- fun compose(f, g) = fn x => f(g(x));
> val ('a, 'b, 'c) compose = fn : ('a -> 'b) * ('c -> 'a) -> 'c -> 'b
```