Solution to Exercise 4.8.4

We put $h := (1,2,3) \in Q$. To compute induced matrix representations we need a straightforward GAP-program `convert` to convert a block matrix $A \in (K^{n\times n})^{m\times m}$ into a matrix in $K^{nm\times nm}$:

```
gap> convert := function(A)
>    local n, m, i, j, k, l, mat, row ;
>     m := Length(A); n := Length( A[1][1] ); mat := [];
>     for i in [1..m] do
>       for j in [1..n] do
>         row:=[];
>         for k in [1..m] do
>           for l in [1..n] do
>             Add( row, A[i][k][j][l] );
>           od;
>         od;
>        Add( mat, row );
>       od;
>     od;
>     return( mat );
>    end;;
```

We choose a left transversal T of Q in G and write a GAP-program `induce` which takes a matrix representation

$$\boldsymbol{\delta}\colon Q \to \mathrm{GL}_n(K),\ h^i \mapsto A^i \qquad (A \in \mathrm{GL}_n(K))$$

and computes $\boldsymbol{\delta}^G(g)$ for any $g \in G$ (with respect to T) where K is a commutative ring.

```
gap> G := SymmetricGroup(4);; h := (1,2,3);;
gap> T := List( RightTransversal( G, Subgroup( G, [h] ) ), x -> x^-1 );;
gap>
gap> induce := function( A, T, h, g )
>    local i, j, k, l, mat;
>     mat := [];
>     for i in [1..Length(T)] do
>       mat[i] := List( [1..Length(T)], x -> 0*A );
>       for j in [1..Length(T)] do
>         k := Position( [h,h^2,h^3], T[j]^-1 * g * T[i] );
>           if k <> fail then  mat[i][j] := A^k; fi;
>       od;
>     od;
>     return( convert(mat) );
>    end;;
```

We choose bases for W_1, W_2, W_3 so that these afford the matrix representations given by

$$\boldsymbol{\delta}_1\colon h \mapsto [1], \qquad \boldsymbol{\delta}_2\colon h \mapsto \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad \boldsymbol{\delta}_3\colon h \mapsto \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

We induce up these representations to G and evaluate them on the generators $(1,2)$ and $(1,2,3,4)$ of S_4 starting with the trivial representation $\boldsymbol{\delta}_1$. This yields a permutation module (of dimension 8) which turns out to have 16 submodules:

```
gap> A := [[1]] * Z(3)^0;;
gap> indrep := List( [(1,2), (1,2,3,4)], x -> induce( A, T, h, x ) );;
gap> module := GModuleByMats( indrep , GF(3) );;
gap> bsm := MTX.BasesSubmodules( module );;   Length(bsm);
16
gap> sm := List( bsm , bas -> Submodule( GF(3)^8 , bas ) );;
```

We write a simple GAP-program which, given the set of submodules of a module V, returns a pair $[V_1, V_2]$ of non-trivial submodules such that $V = V_1 \oplus V_2$ with V_1 indecomposable, or $[V]$, if V is indecomposable:

```
gap> split := function( sm )
>    local i,j,k,n, x,y,res,s ,s1,s2;
>     res:=[ sm[Length(sm)] ];;
>     k := Length( sm ); n := Dimension( sm[k] );
>     for i in [1..QuoInt(n,2)] do
>       s1 := Filtered( sm , x -> Dimension(x) = n-i );
>       s2 := Filtered( sm , x -> Dimension(x) = i );
>        if Length( res ) = 1 and s1 <> [] and s2 <> [] then
>         for y in s1 do
>           s := Filtered(s2,  x -> Dimension(Intersection( x, y )) = 0);
>           if s <> [] and Length( res ) = 1 then res := [ s[1], y ]; fi;
>         od;
>        fi;
>     od;
>     return(res);
>    end;;
```

`split` is used for a GAP-program `decompose` computing for a module V as above a list $[V_1, \ldots, V_r]$ of indecomposable submodules such that $V = V_1 \oplus \cdots \oplus V_r$.

```
gap> decompose := function( sm )
>    local done, summands, spl, sm1;
>     done := false;; summands := [ ];; sm1 := ShallowCopy( sm );
>       while done = false do
>         spl := split( sm1 );    Add( summands, spl[1] );
>         if Length( spl ) = 1 then done := true;
>           else sm1 := Filtered( sm1, x -> IsSubspace( spl[2], x) );
>         fi;
>       od;
>     return( summands );
>    end;;
```

We apply this to W_1^G:

```
gap> summands := decompose( sm );; List( summands, Dimension );
```

```
[ 1, 1, 3, 3 ]
gap> basli := List( summands, x -> bsm[Position( sm, x )] );;
gap> mods := List( basli, x -> MTX.InducedActionSubmodule( module, x ) );;
gap> List( mods,  MTX.IsAbsolutelyIrreducible );
[ true, true, true, true ]
gap> List( [mods{[1,2]},mods{[3,4]} ], x -> MTX.IsEquivalent(x[1],x[2]) );
[ false, false ]
```

Thus $W_1^G = X_1 \oplus X_2 \oplus X_3 \oplus X_4$ with simple $\mathbb{F}_3G$-modules $X_1, X_2 = \mathbb{F}_{3G}, X_3, X_4$ of dimensions $1, 1, 3, 3$. We now turn to W_2^G:

```
gap> A := [ [1,0], [1,1] ] * Z(3)^0;;
gap> indrep := List( [(1,2), (1,2,3,4)], x -> induce( A, T, h, x ) );;
gap> module:=GModuleByMats( indrep, GF(3) );;
gap> bsm := MTX.BasesSubmodules( module );; Length(bsm);
468
gap> sm := List( bsm , bas -> Submodule( GF(3)^16 , bas ) );;
gap> summands := decompose( sm );; List( summands, Dimension );
[ 2, 2, 3, 3, 3, 3 ]
gap> basli := List( summands, x -> bsm[Position( sm, x )] );;
gap> mods := List( basli, x -> MTX.InducedActionSubmodule( module, x ) );;
gap> List( mods,  MTX.IsAbsolutelyIrreducible );
[ false, false, true, true, true, true ]
gap> List( mods{[3..6]}, x -> MTX.IsEquivalent( x, mods[3]) );
[ true, true, false, false ]
gap> List( mods{[3..6]}, x -> MTX.IsEquivalent( x, mods[5]) );
[ false, false, true, true ]
gap> List( [1,2], i -> mods[i].generators );
[ [ [ [ Z(3), 0*Z(3) ], [ Z(3)^0, Z(3)^0 ] ],
      [ [ Z(3), 0*Z(3) ], [ 0*Z(3), Z(3)^0 ] ] ],
  [ [ [ Z(3)^0, 0*Z(3) ], [ Z(3), Z(3) ] ],
      [ [ Z(3)^0, 0*Z(3) ], [ 0*Z(3), Z(3) ] ] ] ]
```

Thus

$$W_2^G \cong U_1 \oplus U_2 \oplus X_3 \oplus X_3 \oplus X_4 \oplus X_4$$

with indecomposable $\mathbb{F}_3G$-modules U_1, U_2 affording the representations

$$\begin{aligned} \boldsymbol{\delta}_1' &: \quad (1,2) \mapsto \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}, \qquad (1,2,3,4) \mapsto \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \\ \boldsymbol{\delta}_2' &: \quad (1,2) \mapsto \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix} \qquad (1,2,3,4) \mapsto \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \end{aligned}$$

Clearly $\boldsymbol{\delta}_1', \boldsymbol{\delta}_2'$ are not equivalent.

We can decompose W_3^G in exactly the same way. But since $W_3 \cong_{\mathbb{F}_3Q} \mathbb{F}_3Q$ we see that $W_3^G \cong_{\mathbb{F}_3G} \mathbb{F}_3G$ and we know the answer from Theorem 1.6.24:

$$W_3^G \cong P(X_1) \oplus P(X_2) \oplus X_3 \oplus X_3 \oplus X_3 \oplus X_4 \oplus X_4 \oplus X_4.$$

Clearly $U_1 \cong P(X_1)/\operatorname{Soc}(P(X_1)) \cong \operatorname{Rad}(P(X_2))$ and $U_2 \cong P(X_2)/\operatorname{Soc}(P(X_2)) \cong \operatorname{Rad}(P(X_1))$.