LMS Short Course on Computational Group Theory
Lab session 9
Using GAP package SCSCP for distributed computations

1. Load the package and define for convenience the server and port which you will be using in this session to avoid typing them each time:

```
LoadPackage("scscp"); server := "lovelace.cs.st-andrews.ac.uk"; port := 261XX;
```

where in the last line XX should be the last two digits of your username (for example, if your username is cptc-001 , then use port :=26101, and if it is cptc-013, use port:=26113)

2. Create a group and first try to find its library number locally:

```
a:=(1,2,3); b:=(2,3); G := Group( a, b ); IdGroup( G );
```

3. Now let's identify it using the remote SCSCP server using two of the services from the lecture:

```
EvaluateBySCSCP( "WS_IdGroup", [ G ], server, port ).object;
EvaluateBySCSCP( "GroupIdentificationService", [ [ a,b ] ], server, port ).object;
```

4. Increase the InfoLevel to see the underlying exchange of SCSCP messages:

```
SetInfoLevel( InfoSCSCP, 3 );
EvaluateBySCSCP( "GroupIdentificationService", [ [ a,b ] ], server, port ).object;
```

5. Create a remote copy of M24 and work with it:

```
M24:=EvaluateBySCSCP("MathieuGroup", [24], server, port :output:="cookie").object;
EvaluateBySCSCP( "NrConjugacyClasses", [M24], server, port );
SetInfoLevel( InfoSCSCP, 3 );
P2:=EvaluateBySCSCP("SylowSubgroup", [M24,2], server, port :output:="cookie").object;
SetInfoLevel( InfoSCSCP, 0 );
RetrieveRemoteObject( P2 );
UnbindRemoteObject( M24 );
```

6. Compare XML and binary OpenMath encodings:

```
x:=SL(2,2); l:=OMString(x); Length(l);
SetInfoLevel( InfoSCSCP, 4 ); SwitchSCSCPmodeToBinary();
x = EvaluateBySCSCP("Identity",[x],server,port).object;
SwitchSCSCPmodeToXML();
```

7. Perform experiments to compare the performance with various encodings. First create the test object:

```
x := [ Z(3)^0, Z(3), 0*Z(3) ]; OMString(x);
for i in [ 4 .. 5000 ] do x[i] := i*Z(3)^0; od; Length(x);
```

Now compare the time of transferring the object to the server and back in four cases:

```
SwitchSCSCPmodeToXML();
x = EvaluateBySCSCP("Identity",[x],server,port).object;
time;
```

```
SwitchSCSCPmodeToBinary();
x = EvaluateBySCSCP("Identity",[x],server,port).object;
time;

SwitchSCSCPmodeToXML();
x = IO_UnpickleFromString( EvaluateBySCSCP( "IO_UnpickleStringAndPickleItBack",
    [ IO_PickleToString(x) ], server, port ).object );
time;

SwitchSCSCPmodeToBinary();
x = IO_UnpickleFromString( EvaluateBySCSCP( "IO_UnpickleStringAndPickleItBack",
    [ IO_PickleToString(x) ], server, port ).object );
time;

SwitchSCSCPmodeToXML();
```

8. Define a pool of two parallel workers:

```
SCSCPservers:= [ [ server, port ], [server, port+40] ];
```

9. Compare the performance of the sequential and parallel versions here. Which is faster? Is that what you expect? Try to explain what you observed.

```
List([1..100],Phi);
ParListWithSCSCP([1..100],"WS_Phi");
```

10. For the next example, where parallelising is efficient, we will need a timing function:

```
Measure:=function(t1,t2)
return 1000000*(t2.tv_sec-t1.tv_sec)+t2.tv_usec-t1.tv_usec;
end;
```

Then, create the list of inputs for the next stage:

```
n:=64; l := List( [ 1 .. NrSmallGroups(n) ], i -> [n,i] );;
```

and run the following command (make sure you input it as a single line):

```
t1:=IO_gettimeofday();ParListWithSCSCP(l,"QuillenSeriesByIdGroup");t2:=IO_gettimeofday();
```

which computes the number of conjugacy classes of elementary abelian subgroups of each rank for each of the groups of order 64. Now the time in microseconds can be calculated as follows:

```
Measure(t1,t2);
```

You may repeat this calculation several times and calculate the average runtime. Now we are suggesting to limit your pool of SCSCP servers to just one server:

```
SCSCPservers:= [ [ server, port ] ];
```

and repeat measurements to see if there is any speedup with using two SCSCP servers instead of one.