```
// import the SCSCPClient and OpenMath libraries
import info.kwarc.mmt.odk.SCSCP.Client.SCSCPClient
import info.kwarc.mmt.odk.OpenMath._

// establish a connection
val client = SCSCPClient("scscp.gap-system.org")

// get a list of supported symbols
/**
 * List(OMSymbol(Size,scscp_transient_1,None,None),
 * OMSymbol(Length,scscp_transient_1,None,None),
 * OMSymbol(LatticeSubgroups,scscp_transient_1,None,None),
 * OMSymbol(NrConjugacyClasses,scscp_transient_1,None,None),
 * OMSymbol(AutomorphismGroup,scscp_transient_1,None,None),
 * OMSymbol(Multiplication,scscp_transient_1,None,None),
 * OMSymbol(Addition,scscp_transient_1,None,None),
 * OMSymbol(IdGroup,scscp_transient_1,None,None),
 * ...,
 * OMSymbol(NextUnknownGnu,scscp_transient_1,None,None))
 */
println(client.getAllowedHeads)


// We make a simple example: Apply the identity function to an integer 1
val identitySymbol = OMSymbol("Identity","scscp_transient_1", None, None)
val identityExpression = OMApplication(identitySymbol, List(OMInteger(1, None)), None, None)
/**
 * OMInteger(1,None)
 */
println(client(identityExpression).fetch().get)

// We also try to compute 1 + 1
val additionSymbol = OMSymbol("Addition", "scscp_transient_1", None, None)
val additionExpression = OMApplication(additionSymbol, OMInteger(1, None) :: OMInteger(1, None) :: Nil, None,
None)

/**
 * OMInteger(2,None)
 */
println(client(additionExpression).fetch().get)

// and close the connection
client.quit()
```