```
using ITensors
using LinearAlgebra

function transfer_matrix_entropy(psi::MPS, region)
  # Construct the reduced transfer matrix
  T = ITensors.transfermatrix(psi, psi, region)

  # Diagonalize the reduced transfer matrix
  evals, evecs = eigen(Matrix(T))

  # Compute the entanglement entropy
  return -sum(abs2.(evals) .* log.(abs2.(evals)))
end

function bose_hubbard_dmrg(N::Int, t::Float64, U::Float64, μ::Float64,
max_bosons::Int, max_sweeps::Int, m::Int)
  # ... [The previous bose_hubbard_dmrg function code] ...

  # Perform DMRG sweeps
  energy, psi = dmrg(H, psi0, sweeps)

  # Calculate observables
  println("Ground state energy: ", energy)

  for j in 1:N
    n = real(expect("N", psi, j))
    println("Site $j occupation number: ", n)
  end

  # Calculate entanglement entropy for various subsystem sizes
  subsystem_sizes = 1:N÷2
  entropies = Float64[]
  for x in subsystem_sizes
    region = 1:x
    S = transfer_matrix_entropy(psi, region)
    push!(entropies, S)
  end

  # Estimate the central charge using the Cardy-Calabrese formula
  L = Float64(N)
  logs = [(log(2L/π * sin(π*Float64(x)/L))) for x in subsystem_sizes]
  A = hcat(logs, ones(length(logs)))
  c_estimate, _ = A \ entropies
  c_estimate *= 6

  println("Estimated central charge: ", c_estimate)
end

# Example usage:
N = 10 # Number of lattice sites
```

```
t = 1.0 # Hopping amplitude
U = 2.0 # On-site interaction energy
μ = 1.0 # Chemical potential
max_bosons = 5 # Maximum number of bosons per site
max_sweeps = 5 # Number of DMRG sweeps
m = 20 # Number of retained states

bose_hubbard_dmrg(N, t, U, μ, max_b
```