```
include("hw4_helpers.jl");

const λ = 1.0;
const a = 1.0;

asoln(x, t) = sin(π * (x - t));

# debugging flag
const test_with_gauss_elim = false;

for h in [1/10; 1/20; 1/40]
  println("Crank-Nicolson, h = $h");
  const k = λ * h;
  const aa = -a * λ / 4;
  const bb = 1.0;
  const cc = -aa;

  xs = linspace(-1.0, 1.0, Int(round((2.0) / h)));
  ts = linspace(0.0, 1.0, Int(round(1.0 / k)));
  const M, K = length(xs), length(ts);

  u = zeros(M, K);
  u[:, 1] = map(x -> asoln(x, 0.0), xs);
  # used for debugging
  u_debug = (test_with_gauss_elim) ? zeros(M, K) : zeros(1, 1);
  if test_with_gauss_elim
    u_debug[:, 1] = map(x -> asoln(x, 0.0), xs);
  end

  @time for n in 1:K-1
    # used for debugging
    A_debug = (test_with_gauss_elim) ? zeros(M, M) : zeros(1, 1);
    b_debug = (test_with_gauss_elim) ? zeros(M) : zeros(1);
    if test_with_gauss_elim
      A_debug[1, 1] = 1.0;
      b_debug[1] = asoln(-1.0, ts[n+1]);
      for m=2:M-1
        A_debug[m, m-1] = aa;
        A_debug[m, m] = bb;
        A_debug[m, m+1] = cc;
        b_debug[m] = u_debug[m, n] - cc * u_debug[m+1, n] - aa * u_debug[m-1, n];
      end
      A_debug[M, M-1] = -λ;
      A_debug[M, M] = 1+λ;
      b_debug[M] = u_debug[M, n];
    end

    # calculate pi and qi for Thomas' algorithm
    p = zeros(M);
    q = zeros(M);
    p[2], q[2] = 0.0, asoln(-1.0, ts[n+1]);
    for m=2:M-1
      dd = u[m, n] - a * λ * (u[m+1, n] - u[m-1, n]) / 4;
      denom = aa * p[m] + bb;
      p[m+1] =  -cc / denom;
      q[m+1] = (dd - aa * q[m]) / denom;
    end
    u[M, n+1] = (u[M, n] + q[M]*λ) / (1 + λ - p[M]*λ);
    for m=M-1:-1:1
      u[m, n+1] = p[m+1] * u[m+1, n+1] + q[m+1];
    end

    if test_with_gauss_elim
      u_debug[:, n+1] = A_debug \ b_debug;
```

```
        @show norm(u[:, n+1] - u_debug[:, n+1], Inf);
      end
    end

    plot_solution(xs, ts, u, asoln; t="Crank-Nicolson, \$h = $h\$",
                  show_plot=false, fname="cn_thomas_M-$M.png");
    if test_with_gauss_elim
      plot_solution(xs, ts, u_debug, asoln; t="Crank-Nicolson, \$h = $h\$",
                    show_plot=false, fname="cn_gauss_M-$M.png");
    end
    u_a = map(x -> asoln(x, 1.0), xs);
    println("Relative L2 error:  ", norm(u[:, end] - u_a, 2) / norm(u_a, 2));
    println("Relative L∞ error:  ", norm(u[:, end] - u_a, Inf) / norm(u_a, Inf));
    println();
  end
```