1. Consider the boundary value problem
\[ u''''(x) = f(x) \equiv 480x - 120, \quad \text{for } x \in (0,1) \] \[ u(0) = u'(0) = u(1) = u'(1) = 0 \]

(a) Derive the following Galerkin formulation for the problem (1)-(2) on some appropriate function space \( V_h \): Find \( u_h \in V_h \) such that
\[ \int_0^1 u_h''(x)v''(x) \, dx = \int_0^1 f(x)v(x) \, dx, \quad \forall v \in V_h. \] \[(3)\]

(b) Define the triangulation \( T_h = \{K_1, K_2\} \), where \( K_1 = [0, \frac{1}{2}] \) and \( K_2 = [\frac{1}{2}, 1] \), and the function space
\[ V_h = \{v \in C^1([0,1]) : v|_K \in P_3(K) \forall K \in T_h, \; v(0) = v'(0) = v(1) = v'(1) = 0\}. \] \[(4)\]

Find a basis \( \{\varphi_i\} \) for \( V_h \). Hint: Consider Hermite polynomials on each element.

(c) Solve the Galerkin problem (3) using your basis functions. Plot the numerical solution \( u_h(x) \) and the true solution \( u(x) \).

2. Implement a Julia function with the syntax
\[ u = fempoi(p,t,e) \]
that solves Poisson’s equation \(-\nabla^2 u(x,y) = 1\) on the domain described by the unstructured triangular mesh \(p,t\). The boundary conditions are homogeneous Neumann (\(n \cdot \nabla u = 0\)) except for the nodes in the array \(e\) which are homogeneous Dirichlet (\(u = 0\)).

Here are a few examples for testing the function:
3. Implement a Julia function with the syntax

\[
\text{errors} = \text{poiconv}(\text{pv, hmax, nrefmax})
\]

that solves the all-Dirichlet Poisson problem for the polygon \( \text{pv} \), using the mesh parameters \( \text{hmax} \) and \( \text{nref} = 0, 1, \ldots, \text{nrefmax} \). Consider the solution on the finest mesh the exact solution, and compute the max-norm of the errors at the nodes for all the other solutions (note that this is easy given how the meshes were refined – the common nodes appear first in each mesh). The output \( \text{errors} \) is a vector of length \( \text{nrefmax} \) containing all the errors.

Test the function using the commands below, which makes a convergence plot and estimates the rates:

```julia
hmax = 0.15
pv_square = Float64[0 0; 1 0; 1 1; 0 1; 0 0]
pv_polygon = Float64[0 0; 1 0; .5 .5; 1 1; 0 1; 0 0]

errors_square = poiconv(pv_square, hmax, 3)
errors_polygon = poiconv(pv_polygon, hmax, 3)
errors = [errors_square errors_polygon]

clf()
loglog(hmax ./ [1,2,4], errors)
rates = @. log2(errors[end-1,:]) - log2(errors[end,:])
```

**Code Submission:** Your Julia file needs to define the functions `fempoi` and `poiconv`, with exactly the requested names and input/output arguments, as well as any other supporting functions and variables that are required for your functions to run correctly.