A.A. List the steps that transform a vertex in a mesh to its location on the screen. And which ones happen in the vertex shader?

modeling isometry M
then inverse camera isometry C^{-1}
then projection P
then clipping
then viewport V
then division

A.B. Why do we clip at the near plane?

to make homogeneous division safe
to not render objects behind camera

A.C. After it is determined that a triangle is partially clipped, what happens next?

linearly interpolate in 4D to compute clipping point
viewport and division
emit one or two triangles

B.A. Which major OpenGL 1.5 feature(s) have we used, that are not in OpenGL 1.4?

vertex buffer objects
to store mesh data on GPU

B.B. Which major OpenGL 2.0 feature(s) have we used, that are not in OpenGL 1.5?

shader programs

B.C. Which major OpenGL 3.2 feature(s) have we used, that are not in OpenGL 2.0?

off-screen framebuffers
vertex array objects
(and backward incompatibility, hence GL3W)

C.A. Briefly describe the shadow mapping algorithm.

two rendering passes

on first pass, render from the light into the shadow map...

...which has only a depth channel

on second pass, render scene as usual...

...but looking up fragment in shadow map

...and not lighting it if shadow depth is shallower there

C.B. Where is the following code? What does it do?

```
uniform mat4 viewingS;
out vec4 pFragmentS;
    mat4 scaleBias = mat4(
        0.5, 0.0, 0.0, 0.0,
        0.0, 0.5, 0.0, 0.0,
        0.0, 0.0, 0.5, 0.0,
        0.5, 0.5, 0.5, 1.0);
    vec4 world = modeling * vec4(position, 1.0);
    pFragmentS = scaleBias * viewingS * world;
```

GLSL code

in vertex shader

simulates transformations of vertex relative to light

so that shadow map lookup can happen (using `pFragmentS`) in fragment shader

`scaleBias` transforms NDC cube $[-1, 1]^3$ to texture-depth cube $[0, 1]^3$

```
uniform mat4 viewingS;
out vec4 pFragmentS;
    mat4 scaleBias = mat4(
        0.5, 0.0, 0.0, 0.0,
        0.0, 0.5, 0.0, 0.0,
        0.0, 0.0, 0.5, 0.0,
        0.5, 0.5, 0.5, 1.0);
    vec4 world = modeling * vec4(position, 1.0);
    pFragmentS = scaleBias * viewingS * world;
```

```
uniform mat4 viewingS;
out vec4 pFragmentS;
    mat4 scaleBias = mat4(
        0.5, 0.0, 0.0, 0.0,
        0.0, 0.5, 0.0, 0.0,
        0.0, 0.0, 0.5, 0.0,
        0.5, 0.5, 0.5, 1.0);
    vec4 world = modeling * vec4(position, 1.0);
    pFragmentS = scaleBias * viewingS * world;
```

```
uniform mat4 viewingS;
out vec4 pFragmentS;
    mat4 scaleBias = mat4(
        0.5, 0.0, 0.0, 0.0,
        0.0, 0.5, 0.0, 0.0,
        0.0, 0.0, 0.5, 0.0,
        0.5, 0.5, 0.5, 1.0);
    vec4 world = modeling * vec4(position, 1.0);
    pFragmentS = scaleBias * viewingS * world;
```