```
// the preview may not look correct on some machiens. In order to view it correctly, render the scene first. (F6 on the keyboard.)
// to enable the slight taper on the ribs, set SupportTaper = true;
// set the dimensions.
//the body is considered to be a cube first. Most of the parts are cut out first and minor details are extruded again.
//dimensions of the containing box
boxWidth = 60.5;
boxLength = 58;
boxHeight = 60;
// bottom layer thickness
floorThickness = 1.75;
// width of the "L" part along X direction
LX = 22.5;
// width of the "L" part along Y direction
LY = 22.5;
// main wall thickness
wallThickness = 1.5;
webThickness = 1.5;
webBottomHeight = 5;
SupportTaper = false;

// just a vector rep of the two axis used very often.
yaxis = [0, 1, 0];
zaxis = [0, 0, 1];
MainModule();


module MainModule()
{
    difference()
    {
        difference()
        {
            cube([boxWidth, boxLength, boxHeight]);
        }
        union(){
            taper(); cavityX(); cavityY();
            translate([LX,LY,floorThickness])
            cube([boxWidth-LX, boxLength - LY, boxHeight - floorThickness]);
                }
    }

    // call each rib now
    translate([LX - 2 * wallThickness + 4,wallThickness,0]) rib(30,1.8, 1.5, webBottomHeight, LY);

    translate([(LX - 2 * wallThickness + 4)*0.5 + (boxWidth - webThickness) * 0.5,wallThickness,0]) rib(30,1.8, 1.5, webBottomHeight, LY);
    translate([boxWidth - webThickness,wallThickness,0]) rib(30,1.8, 1.5, webBottomHeight, LY);

    translate([wallThickness, 3 + LY - 2 * wallThickness + 3,0]) rotate(-90, zaxis)rib(30,1.8, 1.5, webBottomHeight, LX);
    translate([wallThickness, (boxLength -webThickness/2)*0.5 + (3 + LY - 2 * wallThickness + 3) * 0.5,0]) rotate(-90, zaxis)rib(30,1.8, 1.5, webBottomHeight, LX);
    translate([wallThickness, boxLength -webThickness/2,0]) rotate(-90, zaxis)rib(30,1.8, 1.5, webBottomHeight, LX);
}

baseCutW=sqrt(pow(boxWidth-LX,2) + pow(boxLength - LY, 2));

// the taper is a little tricky. It cannot be defined using continuous funcions.
module taper()
if (boxLength - LY > boxWidth - LX)
    translate([LX, boxLength, 0])
        rotate(-45, zaxis)
            cube([baseCutW, baseCutW/2, floorThickness]);
else
```

```
    translate([boxWidth, LY, 0])
        rotate(45, zaxis)
            cube([baseCutW/2, baseCutW, floorThickness]);

module cavityX()
    translate([wallThickness, wallThickness, floorThickness])
        cube([boxWidth, LY - 2 * wallThickness, boxHeight+1]);
module cavityY()
    translate([wallThickness, wallThickness, floorThickness])
        cube([LX - 2 * wallThickness, boxLength - 2 * wallThickness, boxHeight]);

//The minor supports within the cavity
// the ribs are made out of square extrudes. Since the sides are a little inclined, we will use difference with slightly tilted cubes of sufficiently larg dimensions.
module rib(cutHeight, cutLevel1, cutLevel2, wbm, LXY)
    // normalize
    translate([webThickness, 0, 0])
    // normalize
    rotate(-90, yaxis)
    difference()
    {
        // extrude now
        linear_extrude(height = webThickness)
        // make a polygon to extude. It starts at the lowest left corner. Embed the box height and wall thickesses within each rib.
        polygon(points=[
        [floorThickness,0], [boxHeight, 0],

        [cutHeight, webThickness], [cutHeight, webThickness*cutLevel2],

        [wbm + floorThickness, webThickness*cutLevel1],
        [wbm + floorThickness, LXY - 2 * wallThickness - webThickness*cutLevel1],

        [cutHeight, LXY - 2 * wallThickness - webThickness*cutLevel2], [cutHeight, LXY - 2 * wallThickness - webThickness],

        [boxHeight, LXY - 2 * wallThickness], [floorThickness, LXY - 2 * wallThickness]]);
        //sum up everything that needs to be subtracted.
        union()
        {
            //this should create an inclined box of sufficiently large dimensions that can taper the rib.
            if (SupportTaper)
            {
                translate([0,0,webThickness*cutLevel2])rotate(1.5, yaxis)cube([boxHeight*2, LX,webThickness*5]);
                translate([0, 0, webThickness])mirror(-zaxis)translate([0,0,webThickness*cutLevel2])rotate(1.5, yaxis)cube([boxHeight*2, LXY,webThickness*5]);
            }
        }
    }
```