```
// the preview may not look correct on some machiens. In order to view it correctly, render the scene first. (F6 on the keyboard.)
// With EnableThread set to true, the render may take a very long time (up to 3 minutes).
$fn = 30;
// dimensions of the bounding rectangle
Width = 19; Height = 45;
EnableThread = false;
WholePiece();
// chamfer height from top to bottom
chamfer = 7;

// distance between top and outermost wall of the big hole.
Dim1 = 8.3;
// distance between top and outermost wall of the small hole.
Dim2 = 7.9;
BigHoleDia = 10.26;
SmallHoleDia = 7.8;
xaxis = [1, 0, 0];
yaxis = [0, 1, 0];

module WholePiece()
difference()
{
    // the actual body is first considered as a cube.
    cube([Width, Width, Height], center = true);
    // sum up all the cutouts
    union()
{
    // translate the cross-section to the top.
    translate([0,0,-Height / 2 + sqrt(Width*Width+Width*Width)/2 - chamfer])
    // rotate the chamfer cross-section
    rotate_extrude()
    // the cross sectional to make the chamfer cutouts
    polygon(points=[[Height + 10,Height + 10], [0,10 + Height], [0,Height], [Height, 0], [10 + Height, 0]]);

    // both the chmfers are same, so, use mirror for the bottom one. the body being mirrored is nothing but the same cone generated in the previous step.
    mirror([0,0,-1])translate([0,0,-Height / 2 + sqrt(Width*Width+Width*Width)/2 - chamfer])
rotate_extrude()
polygon(points=[[Height + 10,Height + 10], [0,10 + Height], [0,Height], [Height, 0], [10 + Height, 0]]);
    // make the cutting holes length over-long for a better display.
    translate([0, Width/2, Height/2 - BigHoleDia/2 - Dim1])rotate(90, xaxis)cylinder(r = BigHoleDia/2, h = Width * 3, center = true);
    translate([-Width/2, 0, -Height/2 + SmallHoleDia/2 + Dim2])rotate(90, yaxis)cylinder(r = SmallHoleDia/2, h = Width * 3, center = true);

    // make both threads and translate them to the outer walls.
    translate([0,0,Height/2 - Dim1/2]) thread(1.5,3,8,1);
    translate([0,0,-Height/2 + Dim2/2]) thread(1.5,3,8,1);

}
}
// makes an extrusion of a spring like structure using small cylinders. This has bad resolution but is much faster than standard threads.
module spring(h, R, r, pitch)
{
    if (EnableThread)
    {
        // slices of 360 degree long path
        slices = 10;
        circum = 2 * PI  * R;
        // length per slice
        lPerSlice = circum / slices + r;
        inclination = atan2(pitch, circum);
        // spread the cylinders all along the spring path
        for (i = [0:360/slices:h/pitch*360-0.1])
```

```
        {
            // rotate-translate kind of thing.
            // each cylinder is going to be rotated about the axis, lifted with rising pitch distance, also rotated for the thread inclination.
            // normalization
            rotate(90, [0, 1, 0])
            //rotate about the axis
            rotate(-i, [1,0,0])
            //lift upwards
            translate([-pitch * i/360,-R,0])
            //inclination along the thread
            rotate(-inclination, [0,1,0])
            //offset from the center
            translate([0,0,-lPerSlice/2])
            //make each cylinder
            cylinder(r = r, h = lPerSlice);
        }
    }
    else
    {
        cylinder(r = R + r, h = h);
    }
}
// wraps the spring module to give thread specific paramters.
module thread(pitch, dia, height, threadDia)
{translate([0,0,-height/2 + threadDia / 2])union(){translate([0,0,-1])spring(h = height + threadDia, R = dia / 2, r = threadDia / 2, pitch = pitch);
        translate([0,0,-threadDia*2])cylinder(r = dia/2, h = height + threadDia*3);
    }}
```