Instituto Superior Técnico

Programação e Computação para Arquitectura – 2016/2017

Segundo Teste – 19/01/2017

Número: _____

Nome: _____

Escreva o seu número em todas as folhas da prova. O tamanho das respostas deve ser limitado ao espaço fornecido para cada pergunta. Se tiver dúvidas de interpretação, faça suposições razoáveis e explicite-as na sua resposta. Pode usar os versos das folhas para rascunho. A prova tem **4** páginas e a duração é de **60 minutos**. A cotação de cada questão encontra-se indicada entre parêntesis. Boa sorte.

1. (1.0) Defina a função `filtra` que, dado um predicado e uma lista de elementos, devolve uma lista com os elementos para os quais o predicado é verdadeiro. Por exemplo:

```
>> filtra(lambda x: x > 3, [1, 2, 3, 4, 5, 4, 3, 2, 1])
[4, 5, 4]
```

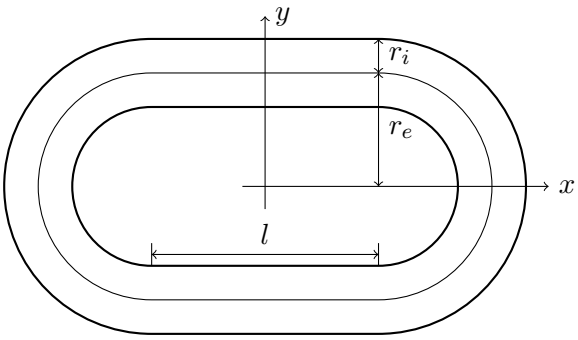
```
def filtra(p, lista):
    if lista == []:
        return []
    elif p(lista[0]):
        return [lista[0]] + filtra(p, lista[1:])
    else:
        return filtra(p, lista[1:])
```

2. (1.0) **Usando** a função `filtra` definida no exercício anterior, defina a função `primeiro_quadrante` que, dada uma lista de posições no plano XY, devolve uma lista contendo apenas as posições localizadas no primeiro quadrante. Por exemplo:

```
>> primeiro_quadrante([xy(-1, 1), xy(1, 2), xy(-2, -1), xy(3, -1), xy(2, 3)])
[xyz(1.0,2.0,0.0), xyz(2.0,3.0,0.0)]
```

```
def primeiro_quadrante(pts):
    return filtra(lambda p: p.x >= 0 and p.y >= 0, pts)
```

3. (1.0) Defina uma função que cria um *elo* de uma *corrente*. Um elo é uma peça de secção circular com as medidas que se apresentam no seguinte esquema:

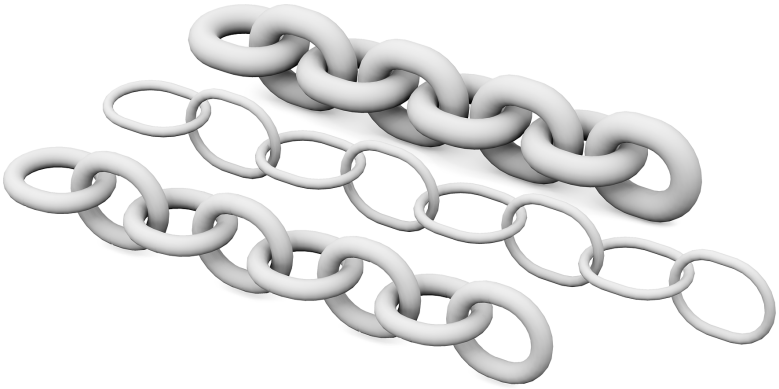


A função `elo` deverá receber, como parâmetros, o raio do elo r_e, o raio do "arame" r_i e o comprimento l entre semi-círculos e deverá criar um elo centrado na origem e orientado como no esquema anterior.

```
def elo(re, ri, l):
    return union(sweep(arc(x(l/2), re, -pi/2, pi),
                       surface_circle(u0(), ri)),
                 cylinder(xy(l/2, re), ri, xy(-(l/2), re)),
                 sweep(arc(x(-l/2), re, pi/2, pi),
                       surface_circle(u0(), ri)),
                 cylinder(xy(-l/2, -re), ri, xy(l/2, -re)))

elo(10, 1, 20)
```

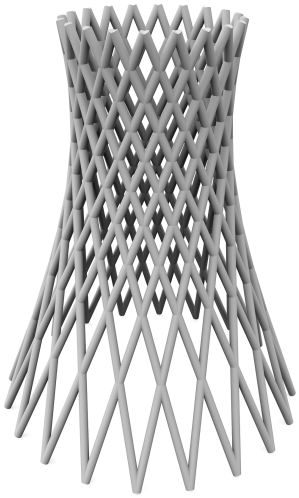
4. (1.0) Defina uma função capaz de criar *correntes* como as que se apresentam em seguida:



Note que, à medida que se vai construindo a corrente, os elos vão sofrendo sucessivas rotações em torno do eixo X. A sua função deverá receber os parâmetros de um elo e ainda o número de elos a criar, e deverá criar a corrente de modo a ter o elo de uma das extremidades centrado na origem.

```
def corrente(n, re, ri, l):
    if n == 1:
        return elo(re, ri, l)
    else:
        return union(elo(re, ri, l),
                     rotate(move(corrente(n - 1, re, ri, l),
                                 vx(l + 2*(re - ri))),
                            pi/2, u0(), vx()))
```

5. (1.0) Considere a *torre hiperbolóide* representada na seguinte imagem à esquerda.

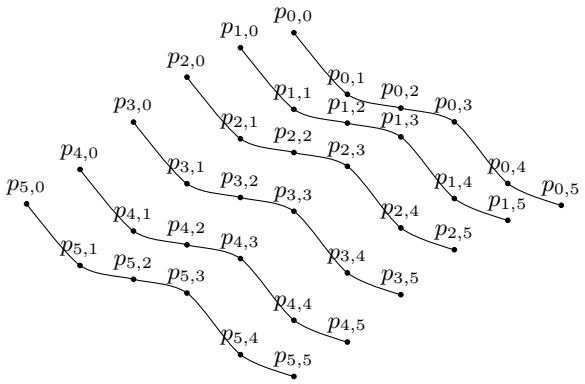


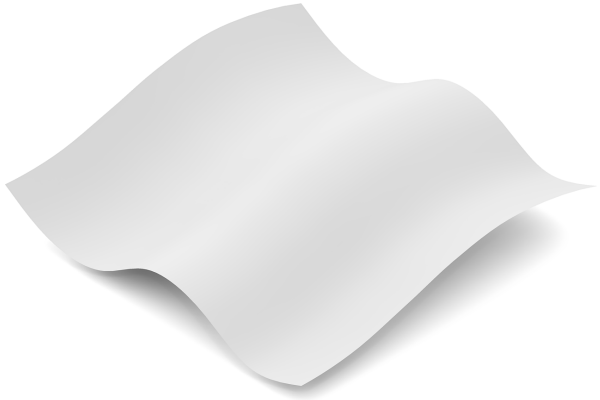
A *torre hiperbolóide* é uma construção que se baseia em dispor $2 \times n$ cilindros de raio r com os topos colocados uniformemente ao longo de duas circunferências, uma na base de raio r_b e outra no topo, de raio r_t e à distância h da primeira, sendo que metade dos cilindros possuem o topo e base desfasados de um ângulo Δ_ϕ e a outra metade de um ângulo de $-\Delta_\phi$, tal como se ilustra na seguinte imagem.
Usando a instrução `for`, defina a função `torre_hiperboloide` que modela a torre a partir do centro da base da torre, dos raios r, r_b e r_t, da altura h, do desfasamento Δ_ϕ e do número n.

```
def torre_hiperboloide(p, r, rb, rt, h, dphi, n):
  for phi in division(0, 2*pi, n, False):
    cylinder(p + vcyl(rb, phi, 0), r, p + vcyl(rt, phi + dphi, h))
    cylinder(p + vcyl(rb, phi, 0), r, p + vcyl(rt, phi - dphi, h))

torre_hiperboloide(xy(1,2), 0.5, 20, 10, 50, pi/2, 20)
```

6. (1.0) Pretende-se produzir a superfície representada na seguinte imagem à direita, a partir de um conjunto de coordenadas tri-dimensionais, tal como se apresenta na imagem à esquerda:





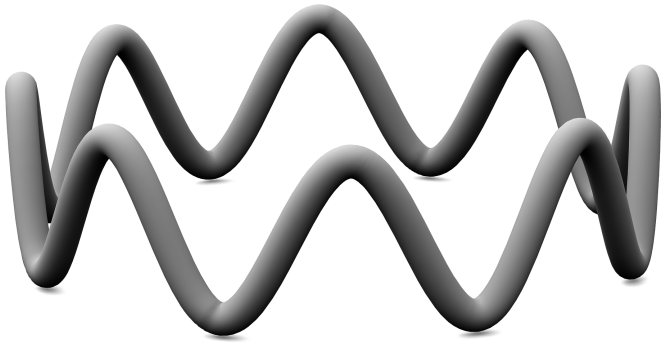
Admita que essas coordenadas estão armazenadas numa lista de listas da forma:

$$
\begin{array}{l}
[\,[p_{0,0}\ \ p_{0,1}\ \ \ldots\ \ p_{0,5}\,] \\
\ [p_{1,0}\ \ p_{1,1}\ \ \ldots\ \ p_{1,5}\,] \\
\ \ldots \\
\ [p_{5,0}\ \ p_{5,1}\ \ \ldots\ \ p_{5,5}\,]\,]
\end{array}
$$

Defina a função `superficie_interpolacao` que recebe uma lista com a forma anterior, começa por criar uma lista de *splines* em que cada *spline* S_i passa pelos pontos $p_{i,0}, p_{i,1}, \ldots, p_{i,5}$ tal como se apresenta na imagem anterior à esquerda e acaba por usar essa lista de *splines* $S_0, S_1, \ldots, S_5$ para fazer uma interpolação suave de secções.

```
def superficie_interpolacao(pontos):
    return loft(map(spline, pontos))
```

7. (2.0) Considere o anel sinusoidal representado na imagem seguinte:



O anel é constituído por um arame de raio r_i cujo eixo é uma sinusóide com uma determinada amplitude a e frequência ω que se desenvolve ao longo de uma circunferência de raio r_e centrada num ponto p. Considere que a curva da sinusóide pode ser aproximada por uma *spline* que passa por n pontos pertencentes à sinusóide.

Defina a função `anel_sinusoidal` que, a partir dos parâmetros p, r_e, r_i, a, ω e n, constrói o anel pretendido.

```
def anel_sinusoidal(p, re, ri, a, omega, n):
  return sweep(
    closed_spline(
        map_division(lambda t: p + vcyl(re, t, a*sin(omega*t)),
                     0, 2*pi, n, False)),
    surface_circle(x(0), ri))
```

8. (2.0) Considere a seguinte esfera composta por tubos sinusoidais tangentes à superfície da esfera. Defina uma função que, convenientemente parameterizada, permite criar exemplos destes.

```
def esfera_sinusoidal(p, re, ri, a, omega, n, m):
  return map_division(
    lambda psi: sweep(
      closed_spline(
        map_division(lambda t: p+vsph(
                        re, t, psi+a*sin(psi)*sin(omega*t)),
                      0, 2*pi, n, False)),
      surface_circle(u0(), ri)),
    0.7*re*ri*a, pi-0.7*re*ri*a, m)

esfera_sinusoidal(xy(1,2), 10, 0.3, 0.15, 10, 100, 20)
```