In [1]:
```
%autosave 0
```

```
Autosave disabled
```

Chapter 23 Convolutions and the Unit Step Function. Convolution describes the combining of functions inside an integral, according to a certain definition. Mathematica examples of use of the Convolve function work well, but equations "in the wild" are much less assured of success. A certain artificial substitution, as in Prob 23.1, usually works, but is not wholly satisfactory, due to the need to ignore an artificially generated element accompanying the solution.

23.1 Find $f(x) * g(x)$ when $f(x) = e^{3x}$ and $g(x) = e^{2x}$.

There are some peculiarities in Mathematica and Wolfram Alpha when it comes to the Convolve function. An aproximation can be achieved by using:

!| Convolve[e^(3 t) UnitStep[t], e^(2t) UnitStep[t], t, x] |!

Exact result

$e^{2x}(e^x - 1)\,\theta(x)$

This presumes that the user is willing to ignore the mysterious theta function expressed here. The θ function also appears self-generated in the input table in W|A, for example in the above:

first function	$e^{3t}\,\theta(t)$
second function	$e^{2t}\,\theta(t)$

23.3 Find $f(x) * g(x)$ when $f(x) = x$ and $g(x) = x^2$.

Continuing the off-label use of the function Convolve. The desired output can be obtained by using:

!| Convolve[t UnitStep[t], t^2 UnitStep[t], t, x] |!

Exact result

$\frac{x^4}{12}\,\theta(x)$

The θ function appears here. Again, it appears to have no effect on the solution, and can be considered equal to 1.

23.4 Find $\mathscr{L}^{-1}\left\{\frac{1}{s^2 - 5s + 6}\right\}$ by convolutions.

Rather than attempt a hard to understand delve into convolutions, falling back on the proven inverse transform call seems justified:

!| simplify[inverselaplacetransform[1/(s^2 - 5 s + 6)]] |!

Result

$e^{2t}(e^t - 1)$

Besides answering the current problem, the above quantity can be recognized as the answer to a problem above.

23.5 Find $\mathscr{L}^{-1}\left\{\frac{6}{s^2 - 1}\right\}$ by convolutions.

Again falling back on the rote inverse transform call:

!| simplify[inverselaplacetransform[6/(s^2 - 1)]] |!

Result
$6\sinh(t)$

For some reason the text did not simplify the solution as shown above.

23.6 Find $\mathscr{L}^{-1}\left\{\frac{1}{s(s^2 + 4)}\right\}$ by convolutions.

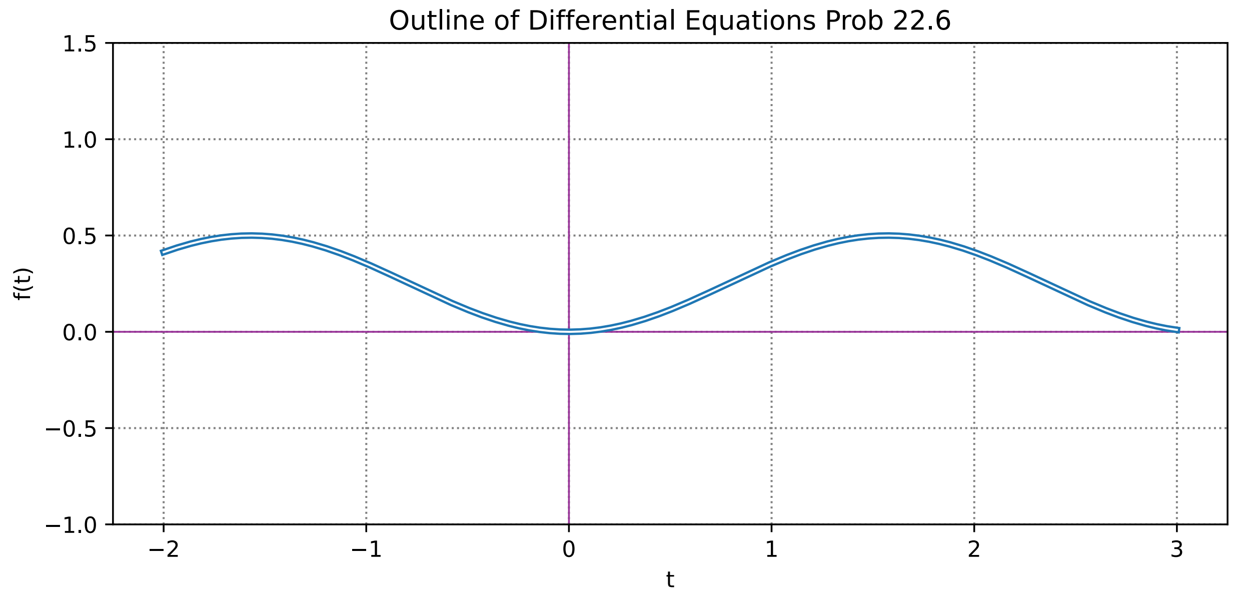
Avoiding the convolutions connection, instead falling back on the proven inverse transform call:

!| simplify[inverselaplacetransform[1/(s (s^2 + 4))]] |!

Result
$\frac{\sin^2(t)}{2}$

Not in the same form as text answer, so a plot will be made to consider the two variations.

In [41]:
```
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

#x = np.arange(0, 3., 0.005)
x = np.linspace(-2,3,300)
y3 = (1/2)*(np.sin(x))*(np.sin(x))
y4 = (1/4)*(1 - np.cos(2*x))


plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("t")
plt.ylabel("f(t)")
plt.title("Outline of Differential Equations Prob 23.6")
plt.rcParams['figure.figsize'] = [9, 7.5]


ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.8)
ax.axvline(x=0, color='#993399', linewidth=0.8)
ratio = 0.95
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)


#plt.text(965, -2.5e43, "-np.log(abs((np.cos(x/2))**2 - (np.sin(x/2))**2)))
#plt.text(965, 2.25e43, "SOLN: y = 3*np.exp(x**2) + 1/2", size=10,\
         # bbox=dict(boxstyle="square", ec=('#8C564B'),fc=(1., 1., 1),))
plt.ylim(-1,1.5)
plt.plot(x, y3, linewidth = 3)
plt.plot(x, y4, linewidth = 0.9, color = 'w')
plt.show()
```



23.7 Find $\mathscr{L}^{-1}\left\{\frac{1}{(s-1)^2}\right\}$ by convolutions.

Once more falling back on the proven inverse transform call seems justified:

!| simplify[inverselaplacetransform[1/((s - 1)^2)]] |!

Result
$e^t\ t$

23.9 Graph the function $f(x) = u(x-2) - u(x-3)$.

This calls for a plot of unit step functions. Seeing that

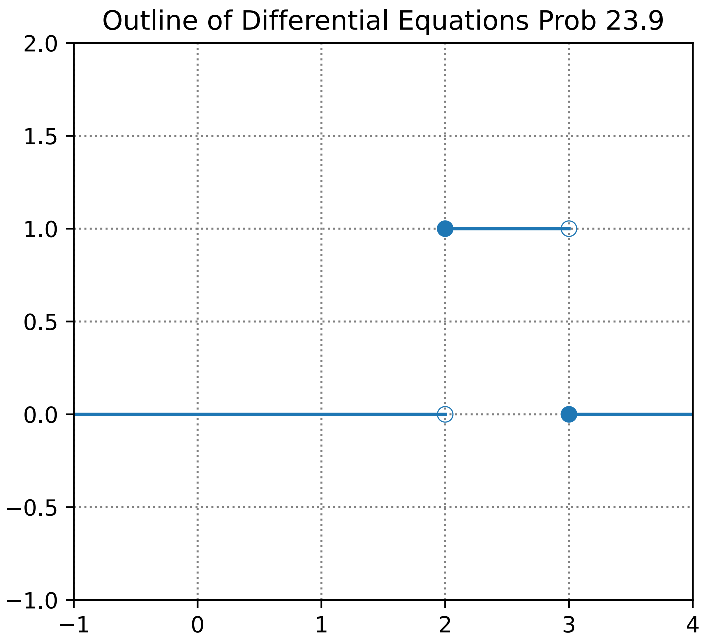
$$u(x-2) = \begin{cases} 0 & x < 2 \\ 1 & x \geq 2 \end{cases} \quad \text{and} \quad u(x-3) = \begin{cases} 0 & x < 3 \\ 1 & x \geq 3 \end{cases}$$

it follows that

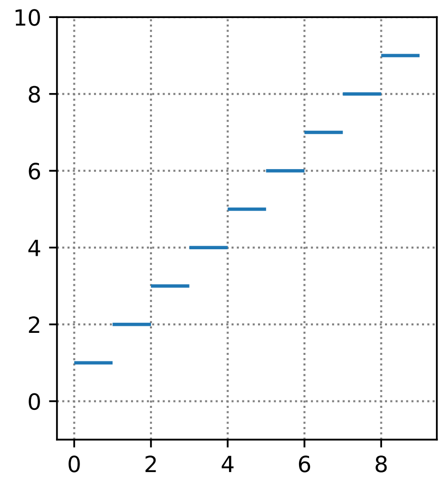
$$f(x) = u(x-2) - u(x-3) = \begin{cases} 0 - 0 = 0 & x < 2 \\ 1 - 0 = 1 & 2 \leq x < 3 \\ 1 - 1 = 0 & x \geq 3 \end{cases}$$

The following plot is a primative attempt to express the above function graphically. But it works. Following the first plot is a second, which does not apply here but is merely a demo.

In [23]:
```
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = [2, 3]
y = [1, 1]
a = [-10, 2]
b = [0, 0]
z = [3, 5]

ax = plt.gca()
plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.rcParams['figure.figsize'] = [5, 5]
ax.set_title('Outline of Differential Equations Prob 23.9')

plt.step(x, y)
plt.step(a, b, color='#1F77B4')
plt.step(z, b, color='#1F77B4')

ratio = 0.1
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

xpts = np.array([2, 3])
ypts = np.array([0, 1])
wpts = np.array([2, 3])
zpts = np.array([1, 0])
plt.plot(xpts, ypts, markersize=7, color='#1F77B4', marker='o', \
         linestyle='none', mfc='none', markeredgewidth=0.5)
plt.plot(wpts, zpts, markersize=7, color='#1F77B4', linestyle ='none', \
         marker='o', mfc='#1F77B4', markeredgewidth=0.5)

plt.ylim(-1,2)
plt.xlim(-1,4)
plt.show()
```



In [63]:
```
import matplotlib.pyplot as plt

y = range(1, 10)
xmin = range(9)
xmax = range(1, 10)

ax = plt.gca()
plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.rcParams['figure.figsize'] = [5, 3.5]

ratio = 1.0
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

plt.ylim(-1,10)
ax.hlines(y, xmin, xmax)
plt.show()
```



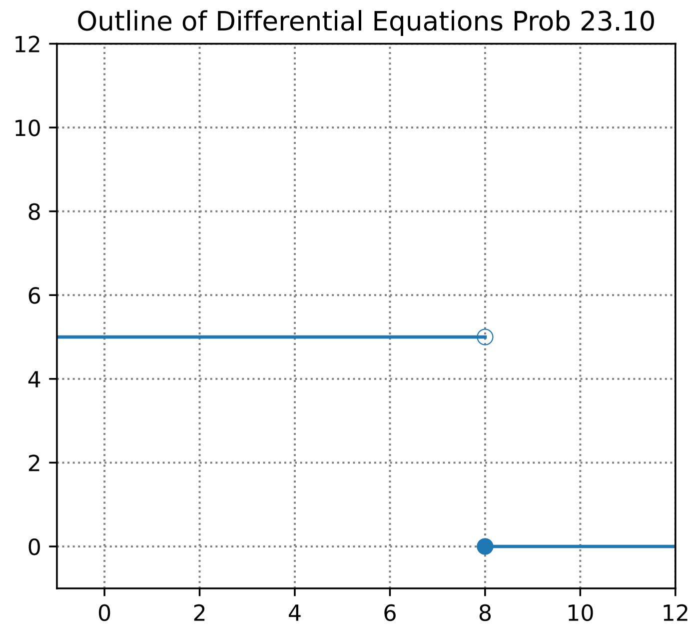
23.10 Graph the function $f(x) = 5 - 5u(x - 8)$ for $x \geq 0$.

This calls for a plot of unit step functions. The function $f(x)$ is

$$f(x) = 5 - 5u(x - 8) = \begin{cases} 5 & x < 8 \\ 0 & x \geq 8 \end{cases}$$

A rough imitation of a stepwise plot is shown.

In [15]:
```
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = [-10, 8]
y = [5, 5]
a = [8, 12]
b = [0, 0]
#z = [3, 5]

ax = plt.gca()
plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.rcParams['figure.figsize'] = [5, 5]
ax.set_title('Outline of Differential Equations Prob 23.10')

plt.step(x, y)
plt.step(a, b, color='#1F77B4')
#plt.step(z, b, color='#1F77B4')

ratio = 0.2
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

xpts = np.array([-5, 8])
ypts = np.array([5, 5])
wpts = np.array([8, 13])
zpts = np.array([0, 0])
plt.plot(xpts, ypts, markersize=7, color='#1F77B4', marker='o', \
         linestyle='none', mfc='none', markeredgewidth=0.5)
plt.plot(wpts, zpts, markersize=7, color='#1F77B4', linestyle ='none', \
         marker='o', mfc='#1F77B4', markeredgewidth=0.5)

plt.ylim(-1,12)
plt.xlim(-1,12)
plt.show()
```



23.11 Find the Laplace transform of $f(x) = \begin{cases} 0 & x < 4 \\ (x-4)^2 & x \geq 4 \end{cases}$.

This problem can be entered into Wolfram Alpha:

!| laplace transform of Piecewise[{{0, x < 4}, {(x - 4)^2, x >= 4}}] |!

In Wolfram Alpha the result is expressed as:

Result

$$\frac{2\,e^{-4p}}{p^3}$$

23.14 Find the Laplace transform of $f(x) = \begin{cases} 0 & x < 4 \\ (x-4)^2 & x \geq 4 \end{cases}$.

This problem can be entered into Wolfram Alpha:

!| laplace transform of Piecewise[{{0, x < 4}, {x^2, x >= 4}}] |!

In Wolfram Alpha the result is expressed as:

Result

$$\frac{2\,e^{-4p}(8p^2+4p+1)}{p^3}$$

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: