We route four different commodities $a, b, x, y$ through the pipes. No commodity can ever flow against the indicated arc direction. Different commodities can be sent through the same pipe. For each arc, the total amount of all commodities that can flow through this pipe is bounded by 10 barrels.

The nodes that are labelled with $A$ can produce arbitrary amounts of commodity $a$. The node that is labelled with $B$ can produce arbitrary amounts of commodity $b$.

At each node that is not labelled with either $A$, $B$, $C$, or $F$, the amount of incoming commodity $a$ equals the amount of outgoing commodity $a$. The same is true for the commodities $b, x, y$.

Nodes labelled with an $F$ are called *factories*. Flow is forwarded in the same way as for other nodes, but a factory can convert commodities as follows before forwarding them.

- To produce 1 barrel of commodity $x$, a factory uses up 2 barrels of $a$ and 1 barrel of $b$.
- To produce 1 barrel of commodity $y$, a factory uses up 1 barrel of $a$ and 3 barrels of $b$.

Factories are not limited to integer barrel amounts. Moreover, factories are not limited to only one way of conversion. For example, a factory can use 2 barrels of $a$ and 3.5 barrels of $b$ to produce 0.5 barrels of $x$ and 1 barrel of $y$.

Nodes labelled with $C$ are called *customers*. A customer pays £2 for a barrel of commodity $x$, and £3 for a barrel of commodity $y$. Customers buy as much of commodity $x$ and $y$ as they can. Fractional barrel amounts are acceptable. Customers never buy commodity $a$ or $b$.

The optimisation problem is maximise the amount of money that the customers pay.

- Model this problem as a linear program (LP) and submit the .lp file. [3 marks]
- Use the *gurobi* software to find an optimal solution and submit the getVars() output as a .txt file. [1 mark]
- Present your solution in a style similar to the way that the example on the next page is presented. A scan of a handwritten submission will also be accepted. Submit your solution as a .pdf file. [1 mark]
It is readily verified that this solution satisfied all constraints. In particular, on each arc the sum of the amounts of commodities never exceeds the arc capacity 10. The customers in this example pay $1 \cdot 2 + 0.5 \cdot 2 + 1 \cdot 3 = 6$ pounds.

**A lengthy, but direct solution.**

We give names to the nodes and more importantly to the 15 arcs:

For each arc we introduce 4 variables that represent how much of the 4 commodities is sent through that arc (for many of them we actually know that they are zero, but we are not interested in making our LP as small as possible here). The LP contains the obvious flow conservation and capacity constraints. The interesting parts are the factory nodes. They are handled as follows:
Let \( v \) be a factory node. Let \( A_{in} \) be the set of arcs pointing into \( v \) and let \( A_{out} \) be the set of arcs pointing out of \( v \). For the sake of a simplified notation, let \( a_{in} := \sum_{e \in A_{in}} a(e) \), and analogously for \( a_{out}, b_{in}, b_{out}, x_{in}, x_{out}, y_{in}, y_{out} \). We make a base change to 8 quantities that describe the factories behaviour. They are all nonnegative (for 4 of them, nonnegativity is already implied by the nonnegativity constraints on the arcs):

- \( a_{used} := a_{in} - a_{out} \) is the amount of \( a \) that is used up to produce commodities at \( v \).
- \( a_{reroute} := a_{in} - a_{used} = a_{out} \) is the amount of \( a \) that is forwarded without being used up at \( v \).
- \( b_{used} := b_{in} - b_{out} \) is the amount of \( b \) that is used up to produce commodities at \( v \).
- \( b_{reroute} := b_{in} - b_{used} = b_{out} \) is the amount of \( b \) that is forwarded without being used up at \( v \).
- \( x_{new} := x_{out} - x_{in} \) is the amount of \( x \) that is produced at \( v \).
- \( x_{reroute} := x_{out} - x_{new} = x_{in} \) is the amount of \( x \) that is forwarded but not created at \( v \).
- \( y_{new} := y_{out} - y_{in} \) is the amount of \( y \) that is produced at \( v \).
- \( y_{reroute} := y_{out} - y_{new} = y_{in} \) is the amount of \( y \) that is forwarded but not created at \( v \).

Instead of the usual 4 flow conservation constraints, at a factory we have 2 degrees of freedom, so we only get 2 equality constraints:

\[
\begin{pmatrix}
2 \\
1 \\
\end{pmatrix}
\begin{pmatrix}
x_{new} \\
y_{new} \\
\end{pmatrix} = \begin{pmatrix}
a_{used} \\
b_{used} \\
\end{pmatrix}.
\]

The LP can be modeled as follows (first constraints are the capacity constraints, then the flow conservation constraints at non-factories, then the equality constraints at factories, and then the nonnegativity constraints at factories).

Maximize
\[
2 \cdot x_{7V12C} + 3 \cdot y_{7V12C} + 2 \cdot x_{9V13C} + 3 \cdot y_{9V13C}
\]
Subject To
\[
\begin{align*}
a_{1A2V} + b_{1A2V} + x_{1A2V} + y_{1A2V} &\leq 10 \\
a_{3V2V} + b_{3V2V} + x_{3V2V} + y_{3V2V} &\leq 10 \\
a_{4F3V} + b_{4F3V} + x_{4F3V} + y_{4F3V} &\leq 10 \\
a_{5V4F} + b_{5V4F} + x_{5V4F} + y_{5V4F} &\leq 10 \\
a_{6A5V} + b_{6A5V} + x_{6A5V} + y_{6A5V} &\leq 10 \\
a_{2V7V} + b_{2V7V} + x_{2V7V} + y_{2V7V} &\leq 10 \\
a_{8F3V} + b_{8F3V} + x_{8F3V} + y_{8F3V} &\leq 10 \\
a_{4F10V} + b_{4F10V} + x_{4F10V} + y_{4F10V} &\leq 10 \\
a_{11B5V} + b_{11B5V} + x_{11B5V} + y_{11B5V} &\leq 10 \\
a_{7V8F} + b_{7V8F} + x_{7V8F} + y_{7V8F} &\leq 10 \\
a_{9V8F} + b_{9V8F} + x_{9V8F} + y_{9V8F} &\leq 10 \\
a_{10V9V} + b_{10V9V} + x_{10V9V} + y_{10V9V} &\leq 10 \\
a_{11B10V} + b_{11B10V} + x_{11B10V} + y_{11B10V} &\leq 10 \\
a_{7V12C} + b_{7V12C} + x_{7V12C} + y_{7V12C} &\leq 10 \\
a_{9V13C} + b_{9V13C} + x_{9V13C} + y_{9V13C} &\leq 10 \\
a_{1A2V} + a_{3V2V} - a_{2V7V} &\leq 0 \\
a_{1A2V} + b_{3V2V} - b_{2V7V} &\leq 0 \\
x_{1A2V} + x_{3V2V} - x_{2V7V} &\leq 0 \\
y_{1A2V} + y_{3V2V} - y_{2V7V} &\leq 0 \\
a_{4F3V} + a_{8F3V} - a_{3V2V} &\leq 0 \\
b_{4F3V} + b_{8F3V} - b_{3V2V} &\leq 0 \\
x_{4F3V} + x_{8F3V} - x_{3V2V} &\leq 0 \\
y_{4F3V} + y_{8F3V} - y_{3V2V} &\leq 0 \\
a_{6A5V} + a_{11B5V} - a_{5V4F} &\leq 0
\end{align*}
\]
\begin{align*}
\text{b6A5V} + \text{b11B5V} - \text{b5V4F} &= 0 \\
\text{x6A5V} + \text{x11B5V} - \text{x5V4F} &= 0 \\
\text{y6A5V} + \text{y11B5V} - \text{y5V4F} &= 0 \\
\text{a4F10V} + \text{a11B10V} - \text{a10V9V} &= 0 \\
\text{b4F10V} + \text{b11B10V} - \text{b10V9V} &= 0 \\
\text{x4F10V} + \text{x11B10V} - \text{x10V9V} &= 0 \\
\text{y4F10V} + \text{y11B10V} - \text{y10V9V} &= 0 \\
\text{a10V9V} - \text{a9V8F} - \text{a9V13C} &= 0 \\
\text{b10V9V} - \text{b9V8F} - \text{b9V13C} &= 0 \\
\text{x10V9V} - \text{x9V8F} - \text{x9V13C} &= 0 \\
\text{y10V9V} - \text{y9V8F} - \text{y9V13C} &= 0 \\
\text{a2V7V} - \text{a7V8F} - \text{a7V12C} &= 0 \\
\text{b2V7V} - \text{b7V8F} - \text{b7V12C} &= 0 \\
\text{x2V7V} - \text{x7V8F} - \text{x7V12C} &= 0 \\
\text{y2V7V} - \text{y7V8F} - \text{y7V12C} &= 0 \\
\text{b1A2V} &= 0 \\
\text{x1A2V} &= 0 \\
\text{y1A2V} &= 0 \\
\text{y6A5V} &= 0 \\
\text{a11B5V} &= 0 \\
\text{x11B5V} &= 0 \\
\text{y11B5V} &= 0 \\
\text{a11B10V} &= 0 \\
\text{x11B10V} &= 0 \\
\text{y11B10V} &= 0 \\
\text{y1B10V} &= 0 \\
\text{a4F3V} + \text{a4F10V} + 2 \text{x4F3V} + 2 \text{x4F10V} + \text{y4F3V} + \text{y4F10V} - \text{a5V4F} &= 0 \\
\text{b4F3V} + \text{b4F10V} + \text{x4F3V} + \text{x4F10V} + 3 \text{y4F3V} + 3 \text{y4F10V} - \text{b5V4F} &= 0 \\
\text{a8F3V} + 2 \text{x8F3V} + \text{y8F3V} - \text{a7V8F} - \text{a9V8F} &= 0 \\
\text{b8F3V} + \text{x8F3V} + 3 \text{y8F3V} - \text{b7V8F} - \text{b9V8F} &= 0 \\
\text{y4F3V} + \text{y4F10V} - \text{y5V4F} &= 0 \\
\text{y5V4F} &= 0 \\

\text{Bounds} \\
\text{a1A2V} &= 0 \\
\text{a3V2V} &= 0 \\
\text{a4F3V} &= 0 \\
\text{a5V4F} &= 0 \\
\text{a6A5V} &= 0 \\
\text{a2V7V} &= 0 \\
\text{a8F3V} &= 0 \\
\text{a4F10V} &= 0 \\
\text{a11B5V} &= 0 \\
\text{a7V8F} &= 0 \\
\text{a9V8F} &= 0 \\
\text{a10V9V} &= 0 \\
\text{a11B10V} &= 0 \\
\text{a7V12C} &= 0 \\
\text{a9V13C} &= 0 \\
\text{b1A2V} &= 0
\end{align*}
b3V2V >= 0
b4F3V >= 0
b5V4F >= 0
b6A5V >= 0
b7V7V >= 0
b8F3V >= 0
b4F10V >= 0
b11B5V >= 0
b7V8F >= 0
b9V8F >= 0
b109W >= 0
b11B10V >= 0
b7V12C >= 0
b9V13C >= 0
x1A2V >= 0
x3V2V >= 0
x4F3V >= 0
x5V4F >= 0
x6A5V >= 0
x2V7V >= 0
x8F3V >= 0
x4F10V >= 0
x11B5V >= 0
x7V8F >= 0
x9V8F >= 0
x10V9W >= 0
x11B10V >= 0
x7V12C >= 0
x9V13C >= 0
y1A2V >= 0
y3V2V >= 0
y4F3V >= 0
y5V4F >= 0
y6A5V >= 0
y2V7V >= 0
y8F3V >= 0
y4F10V >= 0
y11B5V >= 0
y7V8F >= 0
y9V8F >= 0
y109W >= 0
y11B10V >= 0
y7V12C >= 0
y9V13C >= 0

End
The `getVars` output looks as follows.

```
[<gurobi.Var x7V12C (value 0.357142857143)>, <gurobi.Var y7V12C (value 5.71428571429)>, <gurobi.Var x9V13C (value 0.0)>, <gurobi.Var y9V13C (value 0.0)>, <gurobi.Var a1A2V (value 3.92857142857)>, <gurobi.Var b1A2V (value 0.0)>, <gurobi.Var x1A2V (value 0.0)>, <gurobi.Var y1A2V (value 0.0)>, <gurobi.Var a3V2V (value 0.0)>, <gurobi.Var b3V2V (value 0.0)>, <gurobi.Var x3V2V (value 0.357142857143)>, <gurobi.Var y3V2V (value 5.71428571429)>, <gurobi.Var a4F3V (value 0.0)>, <gurobi.Var b4F3V (value 0.0)>, <gurobi.Var x4F3V (value 0.0)>, <gurobi.Var y4F3V (value 2.5)>, <gurobi.Var a5V4F (value 2.5)>, <gurobi.Var b5V4F (value 7.5)>, <gurobi.Var x5V4F (value 0.0)>, <gurobi.Var y5V4F (value 0.0)>, <gurobi.Var a6A5V (value 0.0)>, <gurobi.Var b6A5V (value 0.0)>, <gurobi.Var x6A5V (value 0.0)>, <gurobi.Var y6A5V (value 0.0)>, <gurobi.Var a2V7V (value 3.92857142857)>, <gurobi.Var b2V7V (value 0.0)>, <gurobi.Var x2V7V (value 0.357142857143)>, <gurobi.Var y2V7V (value 5.71428571429)>, <gurobi.Var a8F3V (value 0.0)>, <gurobi.Var b8F3V (value 0.0)>, <gurobi.Var x8F3V (value 0.357142857143)>, <gurobi.Var y8F3V (value 3.21428571429)>, <gurobi.Var a4F10V (value 0.0)>, <gurobi.Var b4F10V (value 0.0)>, <gurobi.Var x4F10V (value 0.0)>, <gurobi.Var y4F10V (value 0.0)>, <gurobi.Var a11B5V (value 0.0)>, <gurobi.Var b11B5V (value 7.5)>, <gurobi.Var x11B5V (value 0.0)>, <gurobi.Var y11B5V (value 0.0)>, <gurobi.Var a7V8F (value 3.92857142857)>, <gurobi.Var b7V8F (value 0.0)>, <gurobi.Var x7V8F (value 0.0)>, <gurobi.Var y7V8F (value 0.0)>, <gurobi.Var a9V8F (value 0.0)>, <gurobi.Var b9V8F (value 10.0)>, <gurobi.Var x9V8F (value 0.0)>, <gurobi.Var y9V8F (value 0.0)>, <gurobi.Var a10V9V (value 0.0)>, <gurobi.Var b10V9V (value 10.0)>, <gurobi.Var x10V9V (value 0.0)>, <gurobi.Var y10V9V (value 0.0)>, <gurobi.Var a11B10V (value 0.0)>, <gurobi.Var b11B10V (value 10.0)>, <gurobi.Var x11B10V (value 0.0)>, <gurobi.Var y11B10V (value 0.0)>, <gurobi.Var a7V12C (value 0.0)>, <gurobi.Var b7V12C (value 0.0)>, <gurobi.Var a9V13C (value 0.0)>, <gurobi.Var b9V13C (value 0.0)>]
```

An illustration of the `gurobi` output looks as follows (decimal digits after the 3rd are truncated).

![Graph](image-url)

The objective function value at the optimum is roughly 17.857.