Part 1: Solve linear program in pg. 5 of section 1.2 of book

```
# Install dependencies
%pip install -q amplpy
```

```
# Google Colab & Kaggle integration
from amplpy import AMPL, ampl_notebook

ampl = ampl_notebook(
    modules=["coin", "highs", "gokestrel"],  # modules to install
    license_uuid="your-license-uuid",  # license to use
)  # instantiate AMPL object and register magics
```

```
AMPL License UUID (you can use free https://ampl.com/ce or
https://ampl.com/courses licenses):
License UUID:
License activated.
Licensed to AMPL Community Edition License for <mavinabeltran@ucdavis.edu>.
```

```
%%ampl_eval
reset;

var XB;
var XC;

maximize Profit: 25*XB + 30*XC;

subject to Time: (1/200)*XB + (1/140)*XC <= 40;

subject to B_limit: 0 <= XB <= 6000;
subject to C_limit: 0 <= XC <= 4000;
```

```
ampl.option["solver"] = "highs"
ampl.solve()
```

```
HiGHS 1.6.0: optimal solution; objective 192000
1 simplex iterations
0 barrier iterations
```

```
ampl.option["solver"] = "gurobi"
ampl.solve()
```

```
Gurobi 10.0.3: optimal solution; objective 192000
0 simplex iterations
```

```
ampl.option["solver"] = "cbc"
ampl.solve()
```

```
cbc 2.10.10: optimal solution; objective 192000
0 simplex iterations


"option abs_boundtol 9.094947017729282e-13;"
or "option rel_boundtol 1.5158245029548803e-16;"
will change deduced dual values.
```

Part 2: Solve same model with upper bound number of bands replaced with 5000

```
%%ampl_eval
reset;

var XB;
var XC;

maximize Profit: 25*XB + 30*XC;

subject to Time: (1/200)*XB + (1/140)*XC <= 40;

subject to B_limit: 0 <= XB <= 5000;
subject to C_limit: 0 <= XC <= 4000;
```

```
ampl.option["solver"] = "highs"
ampl.solve()
```

```
HiGHS 1.6.0: optimal solution; objective 188000
1 simplex iterations
0 barrier iterations
```

```
ampl.option["solver"] = "gurobi"
ampl.solve()
```

```
Gurobi 10.0.3: optimal solution; objective 188000
0 simplex iterations
```

```
ampl.option["solver"] = "cbc"
ampl.solve()
```

```
cbc 2.10.10: optimal solution; objective 188000
0 simplex iterations
```