```
    if header == "ROWS":
      rowIndex = 0
      while lines[i][0] == ' ':
        items = lines[i].split()
        row = Row(items[1].strip(), items[0].strip(), rowIndex)
        if row.type == "N":
          objectiveIndex = rowIndex
        rows[row.label] = row
        rowIndex += 1
        i += 1
        #print row.index, ":", row.label
      tableau = [None] * len(rows)
    elif header == "COLUMNS":
      columnIndex = -1
      while lines[i][0] == ' ':
        items = lines[i].split()
        lim = 2 if len(items) == 5 else 1
        columnLabel = items[0].strip()
        if not columnIndices.has_key(columnLabel):
          columnIndex += 1
          columnLabels.append(columnLabel)
          columnIndices[columnLabel] = columnIndex
        for j in xrange(lim):
          rowLabel = items[1 + j * 2].strip()
          value = float(items[2 + j * 2].strip())
          rows[rowLabel].values[columnLabel] = value
          #print rows[rowLabel].index, ",", columnIndices[
              columnLabel], "=", value
        i += 1
      for j in xrange(len(tableau)):
        tableau[j] = [0] * (len(columnLabels) + 1)
      for row in rows.values():
        #print "row", row.index, ":", len(row.values)
        for colLabel in row.values:
          tableau[row.index][columnIndices[colLabel]] = row.
              values[colLabel]
    elif header == "RHS":
      while lines[i][0] == ' ':
        items = lines[i].split()
        lim = 2 if len(items) == 5 else 1
        for j in xrange(lim):
          rowLabel = items[1 + j * 2].strip()
          value = float(items[2 + j * 2].strip())
          rowIndex = rows[rowLabel].index
          tableau[rowIndex][-1] = value
          #print "RHS of", rowIndex, "=", value
        i += 1
```

```
#print sum([len(r.values) for r in rows.values()])
#for row in tableau:
# tmp = row[-1]
# row[-1] = row[0]
# row[0] = tmp
# print [x for x in row if x != 0]
for row in rows.values():
  tab = tableau[row.index]
  if row.type == "G":
    #print row.index, "is G; multiplying with -1"
    for i in xrange(len(tab)):
      tab[i] = -tab[i]
  elif row.type == "E":
    #print row.index, "is E; creating new row at index ", len(
       tableau)
    tableau.append([-x for x in tab])
#print "objective function is at row", objectiveIndex, ";
   swapping"
tmp = tableau[objectiveIndex]
tableau[objectiveIndex] = tableau[0]
tableau[0] = tmp

ti = 0
while ti < len(tableau):
  nonzero = 0
  for x in tableau[ti]:
    if x != 0:
      nonzero = 1
      break
  if not nonzero:
    tableau.pop(ti)
    ti -= 1
  ti += 1
#print tableau
#for ti in xrange(len(tableau)):
# tab = tableau[ti]
# newTab = []
# for t in tab[:-1]:
#   newTab.append(t)
#   newTab.append(-t)
# newTab.append(tab[-1])
# tableau[ti] = newTab

#tableau[0] = [-x for x in tableau[0]] #for minimisation?

print len(tableau), len(tableau[0])
```

```
for tab in tableau:
  for cell in tab:
    print cell,
  print
sys.exit(0)
print "max: ",
printedAny = 0
for ci in xrange(len(tableau[0]) - 1):
  if tableau[0][ci] != 0:
    if printedAny:
      print " + ",
    printedAny = 1
    print str(tableau[0][ci]) + " x" + str(ci + 1),
print ";"
for tab in tableau[1:]:
  printedAny = 0
  for ai in xrange(len(tab) - 1):
    if tab[ai] != 0:
      if printedAny:
        print " + ",
      printedAny = 1
      print str(tab[ai]) + " x" + str(ai + 1),
  print " <= " + str(tab[-1]) + ";"
for xi in xrange(len(tableau[0]) - 1):
  print "x" + str(xi + 1) + " >= 0;"
```

`cplex.py` — ILOG CPLEX file format parser

```
#!/usr/bin/python
from sys import stdin, stderr

class Equation:
  comparator = ""
  constant = 0
  values = {}

  def __init__(self, comparator, constant):
    self.comparator = comparator
    self.constant = constant
    self.values = {}

class LP:
  pos = 0
  lines = []
  variables = {}
  equations = []
```

```
variableList = []

def __init__(self):
  lines = []
  variables = {}
  equations = []
  variableList = []

def printMatrix(self):
  eqnCount = 0
  for eq in self.equations:
    if eq.comparator == "=":
      eqnCount += 2
    else:
      eqnCount += 1
  print eqnCount, len(self.variables) + 1
  for eq in self.equations:
    line = [0] * (len(self.variables) + 1)
    line[-1] = eq.constant
    for value in eq.values:
      line[self.variables[value]] = eq.values[value]
    negated = [-x for x in line]
    if eq.comparator == "<=" or eq.comparator == "=" or eq.
       comparator == "obj":
      for x in line:
        print x,
      print
    if eq.comparator == ">=" or eq.comparator == "=":
      for x in negated:
        print x,
      print
  for i in xrange(len(self.variableList)):
    stderr.write(str(i + 1) + ": " + self.variableList[i] + "
       \n")

def parseObjective(self):
  tokens = self.lines[self.pos]
  self.pos += 1
  self.parseEquation(tokens, 1)

def parseEquation(self, tokens, isObjective):
  if tokens[1] != '+' and tokens[1] != '-':
    tokens.insert(1, '+')
  if len(tokens) % 2 != 1:
    print "Illegal line length on line", self.pos, ":",
       tokens
  if isObjective:
```

```
      eq = Equation("obj", 0)
    else:
      eq = Equation(tokens[-2], float(tokens[-1]))
    self.equations.append(eq)
    i = 1
    limit = len(tokens) - 1 if isObjective else len(tokens) - 3
    while i < limit:
      if tokens[i] == '-':
        sign = -1
      elif tokens[i] == '+':
        sign = 1
      else:
        print "Illegal sign on line", self.pos, ":", tokens
      name = tokens[i + 1]
      self.addVariable(name)
      eq.values[name] = sign
      i += 2

  def parseEquations(self):
    while 1:
      tokens = self.lines[self.pos]
      if tokens[0][-1] != ':': break
      self.pos += 1
      self.parseEquation(tokens, 0)

  def addVariable(self, name):
    if not self.variables.has_key(name):
      self.variables[name] = len(self.variables)
      self.variableList.append(name)

  def parseBounds(self):
    while 1:
      tokens = self.lines[self.pos]
      if len(tokens) == 1: break
      self.pos += 1
      if len(tokens) == 2 and tokens[1] == "Free":
        pass
      elif len(tokens) == 3:
        #TODO: can the eq be turned around?
        eq = Equation(tokens[1], float(tokens[2]))
        eq.values[tokens[0]] = 1
        self.equations.append(eq)
        self.addVariable(tokens[0])
      elif len(tokens) == 5:
        eq = Equation(">=", float(tokens[0]))
        eq.values[tokens[2]] = 1
        self.equations.append(eq)
```

```
        eq = Equation("<=", float(tokens[4]))
        eq.values[tokens[2]] = 1
        self.equations.append(eq)
        self.addVariable(tokens[2])
      else:
        print "Unrecognised bounds line:", self.pos, ":",
            tokens


  def parse(self):
    for line in stdin:
      tokens = line.split()
      if len(tokens) == 0 or tokens[0] == '\\': continue
      self.lines.append(tokens)
    self.pos = 0
    while self.pos < len(self.lines):
      if self.lines[self.pos][0] == "Maximize":
        self.pos += 1
        self.parseObjective()
      elif self.lines[self.pos][0] == "Subject":
        self.pos += 1
        self.parseEquations()
      elif self.lines[self.pos][0] == "Bounds":
        self.pos += 1
        self.parseBounds()
      else:
        self.pos += 1



lp = LP()
lp.parse()
lp.printMatrix()
```

Appendix B

Test sets

B.1 Sample `netlib` test set

This is the `afiro` set, in MPS format(TODO: citation). It contains (TODO: rows, columns and nonzeroes).

```
NAME          AFIRO
ROWS
 E  R09
 E  R10
 L  X05
 L  X21
 E  R12
 E  R13
 L  X17
 L  X18
 L  X19
 L  X20
 E  R19
 E  R20
 L  X27
 L  X44
 E  R22
 E  R23
 L  X40
 L  X41
 L  X42
 L  X43
 L  X45
 L  X46
 L  X47
 L  X48
 L  X49
```

```
 L  X50
 L  X51
 N  COSTCOLUMNS
    X01       X48              .301   R09              -1.
    X01       R10             -1.06   X05               1.
    X02       X21              -1.    R09               1.
    X02       COST             -.4
    X03       X46              -1.    R09               1.
    X04       X50               1.    R10               1.
    X06       X49              .301   R12              -1.
    X06       R13             -1.06   X17               1.
    X07       X49              .313   R12              -1.
    X07       R13             -1.06   X18               1.
    X08       X49              .313   R12              -1.
    X08       R13              -.96   X19               1.
    X09       X49              .326   R12              -1.
    X09       R13              -.86   X20               1.
    X10       X45             2.364   X17              -1.
    X11       X45             2.386   X18              -1.
    X12       X45             2.408   X19              -1.
    X13       X45             2.429   X20              -1.
    X14       X21               1.4   R12               1.
    X14       COST             -.32
    X15       X47              -1.    R12               1.
    X16       X51               1.    R13               1.
    X22       X46              .109   R19              -1.
    X22       R20              -.43   X27               1.
    X23       X44              -1.    R19               1.
    X23       COST             -.6
    X24       X48              -1.    R19               1.
    X25       X45              -1.    R19               1.
    X26       X50               1.    R20               1.
    X28       X47              .109   R22             -.43
    X28       R23               1.    X40               1.
    X29       X47              .108   R22             -.43
    X29       R23               1.    X41               1.
    X30       X47              .108   R22             -.39
    X30       R23               1.    X42               1.
    X31       X47              .107   R22             -.37
    X31       R23               1.    X43               1.
    X32       X45             2.191   X40              -1.
    X33       X45             2.219   X41              -1.
    X34       X45             2.249   X42              -1.
    X35       X45             2.279   X43              -1.
    X36       X44               1.4   R23              -1.
    X36       COST             -.48
```

```
    X37       X49                -1.   R23                 1.
    X38       X51                 1.   R22                 1.
    X39       R23                 1.   COST               10.
RHS
    B         X50               310.   X51               300.
    B         X05                80.   X17                80.
    B         X27               500.   R23                44.
    B         X40               500.
ENDATA
```

B.2 Test sets provided by Miriam AS

These sets are in the ILOG CPLEX format.

dp_0.lp

```
\Problem name: CPLEX solver

Maximize
 obj: v89_49 + zMax557 + zMax558 + id105
Subject To
 CapE50:      v50_49 - RgCapE50  = 0
 OutBal50_49: v50_49 - x536_49  = 0
 CapE51:      v51_49 - RgCapE51  = 0
 OutBal51_49: v51_49 - x538_49  = 0
 CapE52:      v52_49 - RgCapE52  = 0
 OutBal52_49: v52_49 - x540_49 - x542_49  = 0
 CapS59:      v59_49 - RgCapS59  = 0
 InBal59_49:  x536_49 - v59_49  = 0
 OutBal59_49: v59_49 - x548_49 - x550_49  = 0
 CapS60:      v60_49 - RgCapS60  = 0
 InBal60_49:  x550_49 - v60_49 + x544_49  = 0
 OutBal60_49: v60_49 - x552_49  = 0
 CapS61:      v61_49 - RgCapS61  = 0
 InBal61_49:  x538_49 + x540_49 - v61_49  = 0
 OutBal61_49: v61_49 - x554_49  = 0
 CapS62:      v62_49 - RgCapS62  = 0
 InBal62_49:  x542_49 - v62_49  = 0
 OutBal62_49: v62_49 - x556_49  = 0
 CapI89:      v89_49 - RgCapI89  = 0
 InBal89_49:  x548_49 - x544_49 + x554_49 + x556_49 - v89_49 -
    x546_49 = 0
 NetFlowP89:  - v89_49 + vAbs89 >= 0
 NetFlowN89:  v89_49 + vAbs89 >= 0
 CapD53:      v53_49 - RgCapD53  = 0
```

```
 Dem53_49:    d53_49 <= 150
 Dem53:       d53_49 <= 150
 InBal53_49:  x552_49 - v53_49  = 0
 OutBal53_49: - v53_49 + d53_49  = 0
 CapD54:      v54_49 - RgCapD54  = 0
 Dem54_49:    d54_49 <= 150
 Dem54:       d54_49 <= 150
 InBal54_49:  x546_49 - v54_49  = 0
 OutBal54_49: - v54_49 + d54_49  = 0
 Max557:      - d53_49 - d54_49 + zMax557  = 0
 Comp557:     Comp557  = 0
 SMax558:     - v89_49 + zMax558  = 0
Bounds
 0 <= v50_49 <= 200
 0 <= v51_49 <= 200
 0 <= v52_49 <= 200
 0 <= v53_49 <= 200
 0 <= v54_49 <= 200
      zMax557 >= 299.999
      Comp557 Free
      zMax558 >= -0.001
      id105 = 0
 0 <= RgCapE50 <= 200
 0 <= RgCapE51 <= 200
 0 <= RgCapE52 <= 200
 0 <= RgCapS59 <= 120
 0 <= RgCapS60 <= 200
 0 <= RgCapS61 <= 120
 0 <= RgCapS62 <= 120
 0 <= RgCapI89 <= 900000000
 0 <= RgCapD53 <= 200
 0 <= RgCapD54 <= 200
End
```

dp_150.lp

```
\Problem name: CPLEX solver

Maximize
 obj: v89_49 + zMax557 + zMax558 + id105
Subject To
 CapE50:      v50_49 - RgCapE50  = 0
 OutBal50_49: v50_49 - x536_49  = 0
 CapE51:      v51_49 - RgCapE51  = 0
 OutBal51_49: v51_49 - x538_49  = 0
 CapE52:      v52_49 - RgCapE52  = 0
```

```
 OutBal52_49: v52_49 - x540_49 - x542_49  = 0
 CapS59:      v59_49 - RgCapS59  = 0
 InBal59_49:  x536_49 - v59_49  = 0
 OutBal59_49: v59_49 - x548_49 - x550_49  = 0
 CapS60:      v60_49  = 0
 InBal60_49:  x550_49 - v60_49 + x544_49  = 0
 OutBal60_49: v60_49 - x552_49  = 0
 CapS61:      v61_49  = 0
 InBal61_49:  x538_49 + x540_49 - v61_49  = 0
 OutBal61_49: v61_49 - x554_49  = 0
 CapS62:      v62_49  = 0
 InBal62_49:  x542_49 - v62_49  = 0
 OutBal62_49: v62_49 - x556_49  = 0
 CapI89:      v89_49 - RgCapI89  = 0
 InBal89_49:  x548_49 - x544_49 + x554_49 + x556_49 - v89_49 -
    x546_49  = 0
 NetFlowP89:  - v89_49 + vAbs89 >= 0
 NetFlowN89:  v89_49 + vAbs89 >= 0
 CapD53:      v53_49 - RgCapD53  = 0
 Dem53_49:    d53_49 <= 150
 Dem53:       d53_49 <= 150
 InBal53_49:  x552_49 - v53_49  = 0
 OutBal53_49: - v53_49 + d53_49  = 0
 CapD54:      v54_49 - RgCapD54  = 0
 Dem54_49:    d54_49 <= 150
 Dem54:       d54_49 <= 150
 InBal54_49:  x546_49 - v54_49  = 0
 OutBal54_49: - v54_49 + d54_49  = 0
 Max557:      - d53_49 - d54_49 + zMax557  = 0
 Comp557:     Comp557  = 0
 SMax558:     - v89_49 + zMax558  = 0
Bounds
 0 <= v50_49 <= 100
 0 <= v51_49 <= 100
 0 <= v52_49 <= 100
 0 <= v53_49 <= 200
 0 <= v54_49 <= 200
      zMax557 >= 0.999
      Comp557 Free
      zMax558 >= -0.001
      id105 = 0
 0 <= RgCapE50 <= 200
 0 <= RgCapE51 <= 200
 0 <= RgCapE52 <= 200
 0 <= RgCapS59 <= 1
 0 <= RgCapI89 <= 900000000
 0 <= RgCapD53 <= 200
```

```
 0 <= RgCapD54 <= 200
End
```

Appendix C

Schedule

This appendix will obviously be deleted before submission.

Week 8 Finished the implementation of a dense Simplex for a regular CPU and test with `netlib` datasets. Implement a vectorised (SIMD) dense Simplex on the PPE

Week 9 Struggle with numerical instability

Week 10 Implement a vectorised dense Simplex running in parallel on the SPEs

Week 11 — " — (delayed)

Week 12 — " — (delayed)

Week 13 Study interior point algorithms

Week 14 First draft of report

Week 15 Easter vacation - read on interior point algorithms

Week 16 Implement a dense, non-parallelised interior point algorithm

Week 17 Decide on whether to pursue simplex or interior point. Making a test plan. Experiment with different approaches to sparse storage; look into numerical stability with single-precision values

Week 18 Look into autotuning?

Week 19

Week 20 Performance measurements and graphing

Week 21 Frenetic report writing

Week 22 — " —

Week 23 Ordinary submission deadline. Will try to submit as close to this date as possible

Week 24

Week 25

Week 26

Week 27 Natvig goes on vacation

Week 28

Week 29 Final deadline: July 19