MATH 381 Exam 1

October 27, 2025

Name (as on Gradescope): ____________________

Student #: ____________________

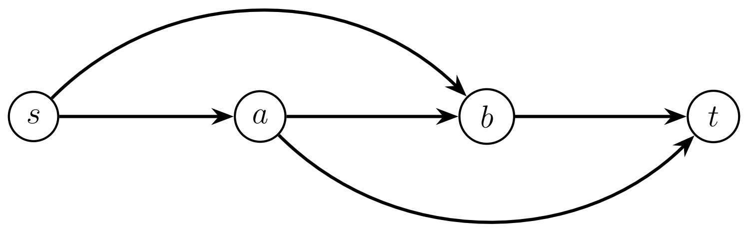
Problem:	1	2	3	Total
Points:	10	9	11	30

INSTRUCTIONS:

- You have 50 minutes to take the test.
- Write your solution below the problem. There is scratch paper at the back of the test.
- If you need extra space to write your solution, use the paper at the back and indicate on which page your solution continues.
- The test is double-sided. Make sure you are reading the backs of pages!
- Unless otherwise stated, show all your work for full credit.
- You are allowed to use one 8.5”×11” sheet of notes, front and back.
- Calculators are not allowed.

Good luck!

1. (10 points) Water is being sent through the following network of one-way pipes, modeled as a directed graph with source s and sink t. The maximum number of gallons of water per minute that each pipe can transfer is shown in the accompanying table.



Pipe	Capacity (gal/min)
s to a	20
s to b	10
a to b	7.5
a to t	10
b to t	18.5

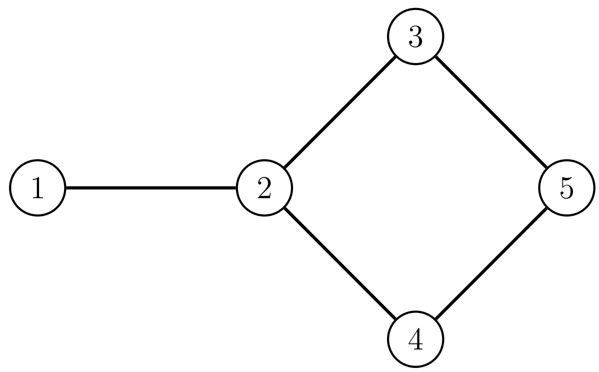
The Python code on the next page begins to construct an OR-Tools solver to find the maximum amount of water this network can transport. Complete the code by filling in the empty spaces in the code. Please **carefully** read the following:

- Not every line of code might be needed. If you think you do not need a line of code, simply do not fill in the blank spaces in that line.
- Do not add extra lines of code.
- The code does not actually solve the problem and print the result; do not add these lines, just leave them omitted.
- You do not need to get the syntax exactly correct (for example, capitalization) but it shouldn't be too far off.

```
from ortools.linear_solver import pywraplp

solver = pywraplp.Solver.CreateSolver("GLOP")

### Create the variables, one for each pipe. ###
x_sa = solver.      Var(      ,      ,"x_sa")
x_sb = solver.      Var(      ,      ,"x_sb")
x_ab = solver.      Var(      ,      ,"x_ab")
x_at = solver.      Var(      ,      ,"x_at")
x_bt = solver.      Var(      ,      ,"x_bt")


### Set the objective. ###
solver.Maximize(                                                    )


### Create the constraints. ###
solver.Add(                                                         )
solver.Add(                                                         )
solver.Add(                                                         )
solver.Add(                                                         )
solver.Add(                                                         )
```

2. (9 points) A company wants to build stores at various locations across a city. A graph model of the city is shown below, with the nodes being the possible store locations. If a store is built at a location, then people from that location as well as each neighboring location will go to that store. For example, if a store is built at location 3, then people from locations 2, 3, and 5 will shop there.

   The company's goal is to build stores so that every location either has a store or is neighboring a location with a store. They want to minimize the number of stores needed to do this.



   The Python code on the next page begins to construct an OR-Tools solver to find the minimum number of stores needed to satisfy the company's goals. Complete the code by filling in the empty spaces in the code. **Unlike the previous problem, you will need to decide the names of the variables yourself.**

   Please recall the following (same as last problem):

   - Not every line of code might be needed. If you think you do not need a line of code, simply do not fill in the blank spaces in that line.
   - Do not add extra lines of code.
   - The code does not actually solve the problem and print the result; do not add these lines, just leave them omitted.
   - You do not need to get the syntax exactly correct (for example, capitalization) but it shouldn't be too far off.

```
from ortools.linear_solver import pywraplp

solver = pywraplp.Solver.CreateSolver("SAT")

### Create the variables. ###
       = solver.      Var(      ,      ,      )
       = solver.      Var(      ,      ,      )
       = solver.      Var(      ,      ,      )
       = solver.      Var(      ,      ,      )
       = solver.      Var(      ,      ,      )


### Set the objective. ###
solver.Minimize(                                                )


### Create the constraints. ###
solver.Add(                                                     )
solver.Add(                                                     )
solver.Add(                                                     )
solver.Add(                                                     )
solver.Add(                                                     )
```

3. Consider a two-player zero-sum game with the following payoff matrix for Player 1.

		P2	
		Strategy C	Strategy D
P1	Strategy A	8	7
	Strategy B	2	4

Player 1 reasons as follows:

- If I play the mixed strategy (x_A, x_B), then my expected payoff will be $8x_A + 2x_B$ if Player 2 plays Strategy C, and $7x_A + 4x_B$ if Player 2 plays Strategy D.
- At equilibrium these two numbers should be equal, so $8x_A + 2x_B = 7x_A + 4x_B$, or $x_A = 2x_B$.
- In addition $x_A + x_B = 1$, so solving the system of equations gives $(2/3, 1/3)$ as my optimal strategy.

(a) (2 points) If Player 1 plays $(2/3, 1/3)$ and Player 2 knows this, what will Player 1's expected payoff be?

(b) (6 points) Find

- an actual optimal strategy for Player 1, and
- the expected payoff for Player 1 at equilibrium (that is, the value of the game).

You can do this by solving a linear program or through another method. There is extra space on the next page.

PROBLEM 3 CONTINUED ON NEXT PAGE.

(c) (3 points) Among the non-pure strategies (that is, mixed strategies (x_A, x_B) where $x_A > 0$ and $x_B > 0$), is $(2/3, 1/3)$ the most optimal? Explain your answer.

END OF EXAM. EXTRA SCRATCH PAPER NEXT.