CS 4050 Test 2 Name ____________

The questions are equally weighted. You may use any notes, books, or handouts.

Note: on all problems, you should write down all your steps, with the only use of a calculator being to do ordinary arithmetic operations. What you write down should give a nicely explained example such as a person would write in a text book. Wherever possible, check your results—trivial errors that should have been caught will be penalized!

---

1. Complete the demonstration of how to efficiently compute a^e in Z_n for $a = 2137$, $e = 37$, and $n = 2501$. Some of the work has been done below, for the standard efficient algorithm—if you don't realize how to use what is given and waste time, that will be unfortunate.

2137	37	
2444	18	
748	9	

2. Complete the demonstration of the extended GCD algorithm for computing the greatest common divisor of a and b, and producing s and t such that $sa + tb$ equals the GCD, with $t > 0$, for $a = 2317$ and $b = 41$.

   Be sure to verify that your final values of s and t do what you want, namely $sa + tb = 1$, with $t > 0$.

a	b	r	q	s	t
2317	41	21	56		
41	21	20	1		

3. Suppose the public information for an RSA encryption scheme is $n = 2501$ and $e = 37$. Suppose further that you intercept the encrypted message $c = 1173$, which you know is a^e in Z_{2501} for the original message a.

   Show all the details to break this encryption.

   Be sure to state the original message a and show all your steps—no credit for just pulling a out of thin air! If you have the energy, you might want to check your answer by encrypting the a you get to verify that this gives you 1173.

   To save some time, here are some powers of c, in Z_{2501}:

c^1	c^2	c^4	c^8	c^{16}	c^{32}	c^{64}	c^{128}	c^{256}	c^{512}
1173	379	1084	2087	1328	379	1084	2087	1328	379

   Note that $2501 = 41 \cdot 61$.

4. Show all the details to compute $9P$ for elliptic curve cryptography, using $p = 13$, using the curve $y^2 = x^3 + 6x + 11$ (i.e., $a = 6$ and $b = 11$), and using the starting point $P = (5, 9)$.

For your convenience, here is the multiplication table for Z_{13}:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	2	4	6	8	10	12	1	3	5	7	9	11
3	0	3	6	9	12	2	5	8	11	1	4	7	10
4	0	4	8	12	3	7	11	2	6	10	1	5	9
5	0	5	10	2	7	12	4	9	1	6	11	3	8
6	0	6	12	5	11	4	10	3	9	2	8	1	7
7	0	7	1	8	2	9	3	10	4	11	5	12	6
8	0	8	3	11	6	1	9	4	12	7	2	10	5
9	0	9	5	1	10	6	2	11	7	3	12	8	4
10	0	10	7	4	1	11	8	5	2	12	9	6	3
11	0	11	9	7	5	3	1	12	10	8	6	4	2
12	0	12	11	10	9	8	7	6	5	4	3	2	1

and here are some results to start the process and save some time:

$1P = (5, 9)$
$2P = (7, 8)$
$4P = (0, 11)$

5. Here is data for an instance of the 0-1 knapsack problem, where the capacity of the knapsack is 10:

j	p_j	w_j
1	126	7
2	80	5
3	90	6
4	56	4
5	24	2
6	33	3

Here is a partially-completed chart for the dynamic programming algorithm to solve this problem, where the rows correspond to the items allowed (in row j you can use any of items 1 through j) and the columns correspond to the weight allowed, and in each cell we have the optimal set of items listed below the optimal profit.

Demonstrate your understanding of the dynamic programming idea for this problem by computing the values for any *needed* cells that are not already filled in—you will be penalized for filling any cells in the bottom three rows that are not needed to obtain the final answer.

Note that all the cells in the first three rows have been filled in, even though some of them were not necessary. The data for the problem is shown off to the left of the chart for your convenience.

After filling in these required cells in the chart, state precisely what choice of items solves this instance of the problem, how much those items weigh, and what profit those items give.

p_j	w_j	j	0	1	2	3	4	5	6	7	8	9	10
126	7	1	0 {}	0 {}	0 {}	0 {}	0 {}	0 {}	0 {}	126 {1}	126 {1}	126 {1}	126 {1}
80	5	2	0 {}	0 {}	0 {}	0 {}	0 {}	80 {2}	80 {2}	126 {1}	126 {1}	126 {1}	126 {1}
90	6	3	0 {}	0 {}	0 {}	0 {}	0 {}	80 {2}	90 {3}	126 {1}	126 {1}	126 {1}	126 {1}
56	4	4											
24	2	5											
33	3	6											

6. Below you will find the matrix $D^{(4)}$ obtained part of the way through doing Floyd's algorithm on a given set of edge weights.

   Do the next step of Floyd's algorithm from this point, computing what cells are improved by allowing vertex 5 to be used as an intermediate vertex (for your convenience on this step, row 5 and column 5 have been highlighted).

   Mark any changes directly in the given diagram, crossing out old values and writing in new ones.

   Be sure to update both numbers in any cell where a change is made.

	1	2	3	4	5	6
1	0 0	12 0	10 0	7 0	2 0	3 0
2	10 0	0 0	14 0	16 0	12 1	12 0
3	15 0	18 0	0 0	19 0	17 1	18 1
4	19 0	21 0	15 0	0 0	17 0	22 1
5	3 0	2 0	4 0	3 0	0 0	6 1
6	2 0	5 0	6 0	1 0	4 0	0 0

Then show how to use the chart to find the optimal path from vertex 2 to vertex 4, assuming that any of vertices 1 through 5 can be used as intermediate vertices. Be sure to show the entire sequence of vertices.

7. Here are the weights for a directed graph and a chart for the dynamic programming approach to the Traveling Salesman Problem on the given graph:

	1	2	3	4	5
1	0	4	7	6	5
2	2	0	3	2	4
3	2	6	0	8	1
4	3	5	1	0	2
5	6	10	4	12	0

Your job on this problem is to clearly show how this dynamic programming information can be used to find an optimal tour for the given graph. Circle all chart items that you use, show all your computations, and clearly state the vertices and weights of each edge in the optimal tour.

```
                  2         3         4         5
        {}  2.00( 0)  2.00( 0)  3.00( 0)  6.00( 0)
       {2} ---------  8.00( 2)  7.00( 2) 12.00( 2)
       {3}  5.00( 3) ---------  3.00( 3)  6.00( 3)
     {2,3} --------- ---------  9.00( 3) 12.00( 3)
       {4}  5.00( 4) 11.00( 4) --------- 15.00( 4)
     {2,4} --------- 11.00( 2) --------- 15.00( 2)
     {3,4}  5.00( 4) --------- --------- 15.00( 3)
   {2,3,4} --------- --------- --------- 15.00( 2)
       {5} 10.00( 5)  7.00( 5)  8.00( 5) ---------
     {2,5} --------- 13.00( 5) 14.00( 5) ---------
     {3,5} 10.00( 3) ---------  8.00( 3) ---------
   {2,3,5} --------- --------- 14.00( 3) ---------
     {4,5} 10.00( 4) 16.00( 4) --------- ---------
   {2,4,5} --------- 16.00( 2) --------- ---------
   {3,4,5} 10.00( 4) --------- --------- ---------
 {2,3,4,5} --------- --------- --------- ---------
```

8. Consider this problem: given a directed, unweighted graph with vertices v_1, v_2, …, v_n, and no cycles, report the number of different paths from v_1 to v_n.

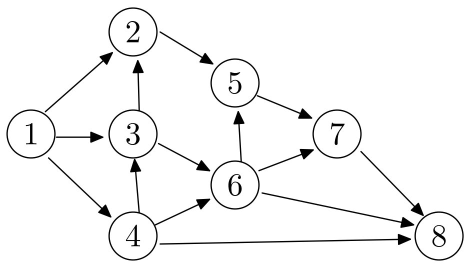
   Your job is to use the dynamic programming approach to design an algorithm for this problem, following the suggested ideas (no credit for inventing your own algorithm following different ideas!).

   Use a one-dimensional chart, where $N[j]$ is the number of different paths from v_1 to v_j. With this idea, $N[1]$ is the number of different paths from v_1 to v_1, which is the base case for the recursion, and is taken as 1.

a. [1 point] What cell in the chart gives the answer to the entire problem?

b. [3 points] Figure out and explain the key recursive idea, namely how can cell $N[j]$ be computed assuming that the required cells $N[k]$ for other vertices v_k have already been computed? Note that your answer to this will need to refer to the graph.

c. [4 points] Demonstrate the full algorithm by filling in the empty dynamic programming chart given below using this graph:



1	2	3	4	5	6	7	8

As part of this calculation, to make sure that you are using the recursive idea and not just brute force counting, state *precisely* how $N[8]$ is computed in terms of other cells of the chart: