HOMEWORK C
DATA STRUCTURES, RECURSION, CODE PROFILING[1]

CMU 10-607: COMPUTATIONAL FUNDAMENTALS FOR MACHINE LEARNING (FALL 2018)

https://piazza.com/cmu/fall2018/10606607

OUT: Nov. 28, 2018

DUE: Dec 3, 2018 11:59 PM

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: http://www.cs.cmu.edu/~mgormley/courses/606-607-f18/about.html#7-academic-integrity-policies
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/606-607-f18/about.html#6-general-policies
- **Submitting your work to Gradescope:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeX users, use ■ and ● for shaded boxes and circles, and don't change anything else.

[1] Compiled on Wednesday 28th November, 2018 at 02:28

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

⊗ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

□ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

⊠ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

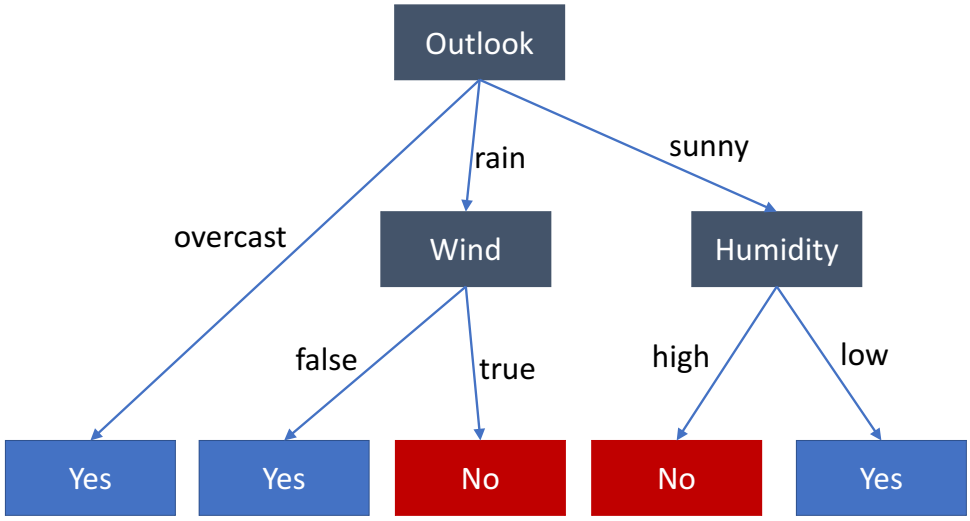
Fill in the blank: What is the course number?

10-606 10-~~7~~06

1 Data Structures and Recursion [2 pts]

Implement a decision tree data structure in Python. The easiest way to do this is to use a Python dictionary where the condition at a node can be used as a string key to traverse to the children of the node.

Using your tree implementation, create the following decision tree in Python code:

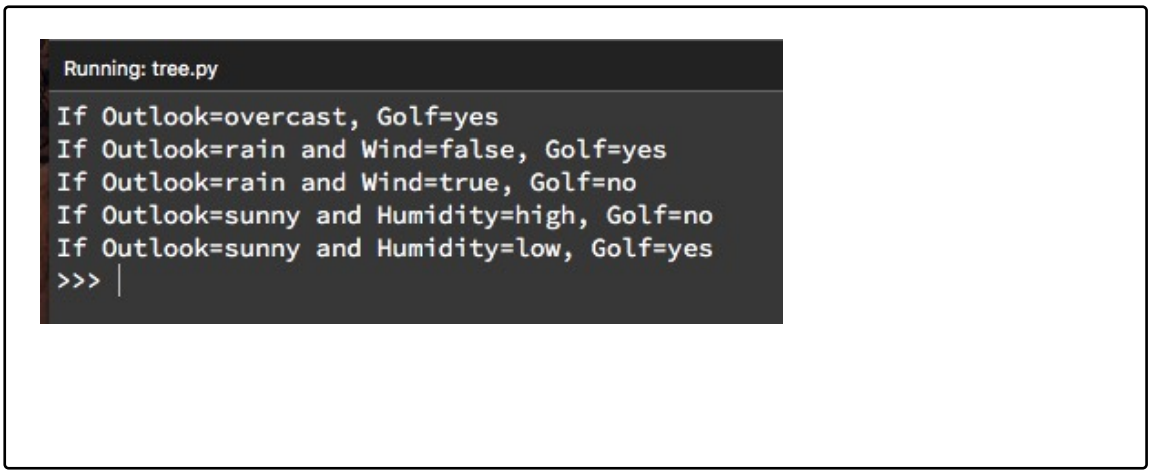


Use depth-first search of the tree to read out each possible rule from the root of the tree to each leaf in the tree. The output of your code should resemble the following:

```
If Outlook=overcast, Golf=yes
If Outlook=rain and Wind=false, Golf=yes
If Outlook=rain and Wind=true, Golf=no
If Outlook=sunny and Humidity=high, Golf=no
If Outlook=sunny and Humidity=low, Golf=yes
```

All your code for this question should be in a Python file named `tree.py`. Running the file using the command `python tree.py` should generate the console output provided above.

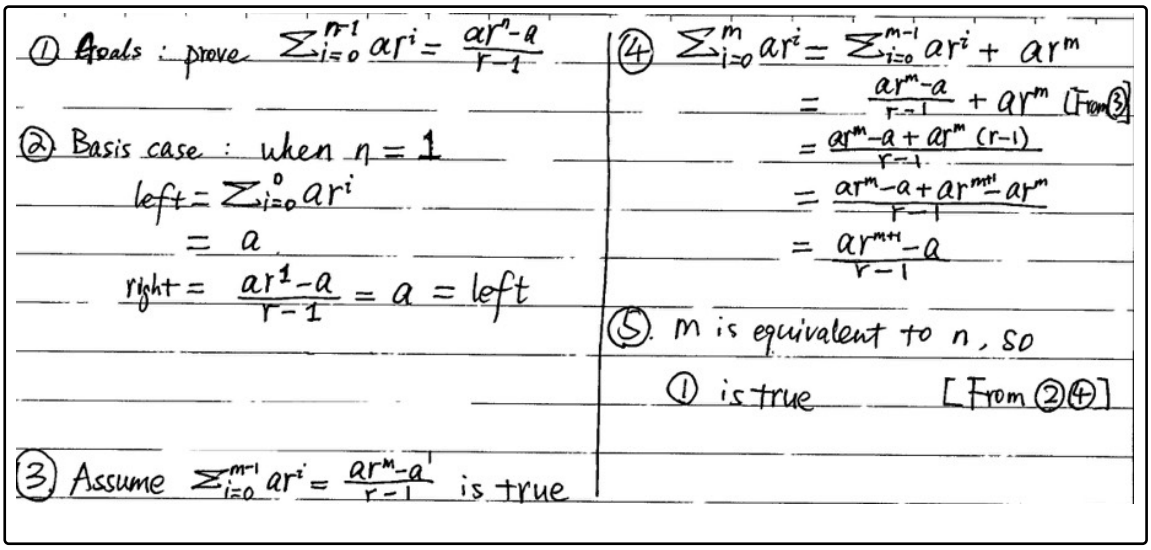
1. **[2 pts]** Include a legible screenshot of your console output here.



2 Proof by Induction [45 pts]

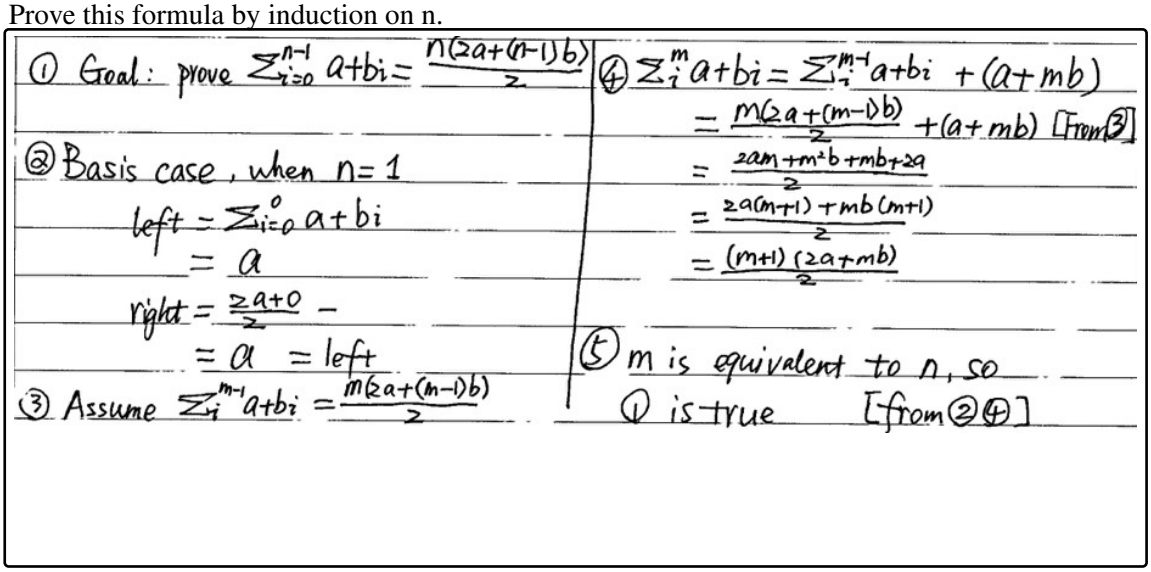
(All problems are from Aho and Ullman Chapters 2, 3. Feel free to consult them and any other resources that are helpful for solving the problems)

1. **[5 pts]** The formula for the sum of n terms of a geometric series a, ar, $ar^2 \ldots ar^{n1}$ is $\sum_{i=0}^{n-1} ar^i = \frac{ar^n - a}{r-1}$ Prove this formula by induction on n. Note that you must assume r $\neq 1$ for the formula to hold.



2. **[5 pts]** The formula for the sum of an arithmetic series with first term a and increment b, that is, a,(a + b),(a + 2b),. . . , a + (n 1)b

$\sum_{i=0}^{n-1} a + bi = \frac{n(2a+(n-1)b)}{2}$.
Prove this formula by induction on n.



3. You are given the algorithm for recursive selection sort as follows:

```
def recSS(A, i, N):
    '''
    @param A: A list of integers
    @param i: start_index
```

```
@param n : length of array
'''
while i < n-1:
    small = i
    for each_j in range(i+1, n):
        if A[j] < A[small]:
            small = j
    temp = A[small]
    A[small] = A[i]
    A[i] = temp
    recSS(A, i+1, n)
```

(a) **[5 pts]** Suppose that we are given array A[0..4], with elements 10, 13, 4, 7, 11, in that order. What are the contents of the array A just before each recursive call to recSS, according to the recursive function specified above.

[4, 13, 10, 7, 11]
[4, 7, 10, 13, 11]
[4, 7, 10, 13, 11]
[4, 7, 10, 11, 13]

(b) **[5 pts]** Prove by induction that the running time of this algorithm is O(n^2).

① Goal: prove incursive selection sort is $O(n^2)$

② Basis case, when n=1
T=1 is $O(n^2)$

③ Assume for m elements,
$T(m) \leq m^2$

④ The time complexity of sort/insert a number into a m-element array is m

⑤ $T(m+1) = T(m) + m$
$\leq m^2+m$
$\leq (m+1)^2$

⑥ So, the running time of the algorithm is $O(n^2)$

4. **[5 pts]** The greatest common divisor (GCD) of two integers i and j is the largest integer that divides both i and j evenly. For example, gcd(24, 30) = 6, and gcd(24, 35) = 1. Write a recursive function that takes two integers i and j, with i $>$ j, and returns gcd(i, j). (Hint: You may use the following recursive definition of gcd. It assumes that i $>$ j. BASIS. If j divides i evenly, then j is the GCD of i and j. INDUCTION. If j does not divide i evenly, let k be the remainder when i is divided by j. Then gcd(i, j) is the same as gcd(j, k).)

```
1  def gcd(i,j):
2      assert i>j
3      k = i%j
4      if k ==0:
5          return j
6      else:
7          return gcd(j,k)
```

5. **[5 pts]** Prove the correctness of the recursive version of GCD.

Goal: prove gcd(a, b) = gcd(b, a mod b)

Subgoal1: prove gcd(a,b) | gcd(b, a mod b)
Let d = gcd(a,b)
Thus d | a, d | b
We have theorem: a mod b = a − b⌊a/b⌋, where b ∈ $\mathbb{Z}^+$
Thus, (a mod b) is a linear combination of a and b.
Thus d | (a mod b) [Theorem]
From d | b and d | (a mod b), we get
d | gcd(b, a mod b), which proves subgoal1.

Subgoal2: prove gcd(b, a mod b) | gcd(a,b)
Let e = gcd(b, a mod b)
Thus e | b and e | a mod b
Since a mod b = a − b⌊a/b⌋,
a mod b is the linear combination of a and b
If e | b⌊a/b⌋, than e | a [Theorem]
From e | b and e | a, we get
e | gcd(a,b),
which proves subgoal2

From subgoal 1 &2, we prove our goal [Theorem]

6. **[5 pts]** Show that the running time of the program is O(log i) (Hint: Show that after two calls we invoke gcd(m, n) where m $\leq \frac{i}{2}$.).

Assume we want to compute gcd (m, n), where m > n.
First call: gcd (m, n) if m mod n equals to 0, n is GCD. This is O(1)
Second call: gcd(n, m mod n) if m mod n does not equal to 0.
Third call: gcd(m mod n, n mod (m mod n))
We assume $(m \bmod n) > \frac{m}{2}$,
thus, $n > \frac{m}{2}$ [divisor is greater than remainder]
thus, $(m \bmod n) < \frac{m}{2}$ $[m/n < m/\frac{m}{2} = 2]$
which is contradict with assumption ①
So, $(m \bmod n) < \frac{m}{2}$
Thus, $T(m) \leq T(\frac{m}{2}) + 2$
Thus, the algorithm is $O(\log i)$

7. **[5 pts]** Write the non-recursive form of the GCD algorithm and prove that it gives the same answer as the recursive form.

When i mod j equals to 0, both of the gcd algorithm will return j.

When i mod j does not equals to 0, both algorithm will replace the nominator with j, and replace the denominator with the reminder (k). Then, both will repeat the division until reminder is 0; they will return the denominator in the invoke/iteration.

```
1  def gcd(i,j):
2      assert i>j
3      k = i%j
4      if k ==0:
5          return j
6      else:
7          return gcd(j,k)
8
9  def gcd_iterative(i, j):
10     assert i>j
11     k = i%j
12     while k != 0:
13         i = j
14         j = k
15         k = i%j
16     return j
```

8. **[5 pts]**

```
class Node:
    def __init__(self, dataval=None):
        self.dataval = dataval
        self.nextval = None

class SLinkedList:
    def __init__(self):
        self.headval = None

list1 = SLinkedList()
list1.headval = Node("A")
elem2 = Node("B")
elem3 = Node("C")
# Link first Node to second node
list1.headval.nextval = elem2

# Link second Node to third node
elem2.nextval = elem3

def printlist(list):
    '''
    @param list: Linked list implemented through list of dictionaries, where ea
    '''
    while list !=None:
        print(list.headval.dataval)
        list.headval = list.headval.next
printlist(list1)
```

Prove that the function PrintList prints the elements on the list that it is passed as an argument. What statement S(i) do you prove inductively? What is the basis value for i?

1) basis case: Since list1 is not None (it points to a SLinkedList object), the while loop will be executed. list.headval is the Node("A"), so the dataval, which is a String "A" will be printed.

2) assume when i = k, k statements (nodes) will be printed.
3) when i = k+1, since k nodes will be printed (from step 2), and the kth node will have a nextval attribute pointing to the (k+1)th node, which is not None. Thus, k+1 elements will be printed.
4) Based on 1) and 3), we prove that the function prints all elements on the list. The basis value for i is 1.
(Note: the 2nd line in while loop should be: list.headval = list.headval.nextval)

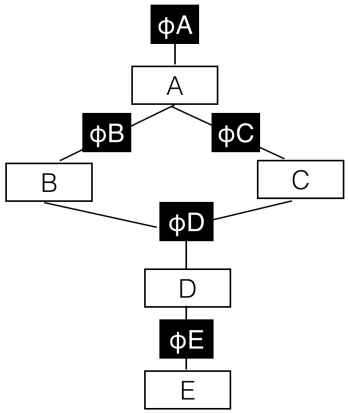


Figure 3.1: Bayes Net depicting variables and their dependencies.

3 Variable Elimination [10 pts]

Figure 2.1 depicts a factor graph for a given Bayes Net, and we are interested in finding the probability distribution P(E). You are given a partial implementation of Variable Elimination for computing marginal probability of P(E) including the factor graph data structure.

Specifically, you are provided with the following classes:

- class Variable This class allows one to define Bayes Net variables. On initialization the variable object can be given a name and a domain of values. This list of domain values can be added to or deleted from in support of an incremental specification of the variable domain. The variable also has a set and get value method. These set a value for the variable that can be used by the factor class.
- class Factor This class allows one to define a factor specified by a table of values. On initialization the variables the factor is over is specified. This must be a list of variables. This list of variables cannot be changed once the constraint object is created. Once created the factor can be incrementally initialized with a list of values. To interact with the factor object one first sets the value of each variable in its scope (using the variable's set_value method), then one can set or get the value of the factor (a number) on those fixed values of the variables in its scope. Initially, one creates a factor object for every conditional probability table in the bayes-net. Then one initializes the factor by iteratively setting the values of all of the factor's variables and then adding the factor's numeric value using the add_value method.
- class BN This class allows one to put factors and variables together to form a Bayes net. It serves as a convient place to store all of the factors and variables associated with a Bayes Net in one place. It also has some utility routines to, e.g,., find all of the factors a variable is involved in.

Please implement the following methods.

1. **[1 pts]** function Naive_VE(Net, QueryVar): Please implement the naive version of the algorithm to find the distribution P(E) ie. compute the joint distribution of all variables for a given query E.
2. **[2 pts]** function get_bfs_ordering(Factors, QueryVar) which uses BFS and provides the order for variable elimination. Provide an assertion within the function to ensure the order is what you expect it should be. Return an ordered list of Variable objects.

3. **[2 pts]** function __collapse(NetFactors, QueryVar): collapsing function that removes the variable X, its neighboring factors and adds in a new factor as appropriate.
4. **[2 pts]** function __variable_elimination(Net, QueryVar). Your final implementation should accept a factor graph and a query variable (i.e. the one for which we want the marginal distribution).
5. What would the appropriate assertions be for the following cases? Please provide your answers here as well as include them within the code.
   (a) **[1 pts]** Assertion about the size of the resulting factor after collapsing
   (b) **[1 pts]** Assertion about the neighbors of the resulting factor after collapsing
   (c) **[1 pts]** Assertion about each factor summing to one.

4 Profiling [9 pts]

1. **Time Profiling**

   In this question you will be profiling code for execution time using cProfile. Profiling your code is often a useful step in understanding your code from a time point of view. As mentioned in the documentation "cProfile and profile provide deterministic profiling of Python programs. A profile is a set of statistics that describes how often and for how long various parts of the program executed.". You can find the documentation here - https://docs.python.org/2/library/profile.html. Cprofile provides the following statistics:

   - ncalls: for the number of calls,
   - tottime: for the total time spent in the given function (and excluding time made in calls to sub-functions)
   - cumtime :is the cumulative time spent in this and all subfunctions (from invocation till exit). This figure is accurate even for recursive functions.
   - percall: is the quotient of cumtime divided by primitive calls
   - filename:lineno(function) provides the respective data of each function

   cProfile can be run using:

   ```
   python -m cProfile [-o output_file] [-s sort_order] myscript.py
   -o writes the profile results to a file instead of to stdout
   ```

   -s specifies one of the sort_stats() sort values to sort the output by. This only applies when -o is not supplied.

   (a) **[2 pts]** We will start by profiling the code you have already written for Question. 3 (Variable Elimination). Profile the time of the function calls to Naive_VE and VE methods. Remember the method Naive_VE is where you have computed the joint probability distribution over all the variables. Report the total time of both functions.

(b) **[2 pts]** You will find file sparse_dot.py (included in the handout) computes dense dot products over a sparse binary vector with a dense one. Using the command 'python -m cProfile -s tottime sparse_dot.py', identify the function which takes the most time in total, and specify both the function and the time taken below.

(c) **[2 pts]** You will find file efficient_sparse_dot.py does sparse computation over the same vectors. Using the command 'python -m cProfile -s tottime efficient_sparse_dot.py', identify the function which takes the most time in total, and specify both the function and the time taken below. Compare it to the total time taken for the same function in sparse_dot.py by specifying the difference.

(d) **[1 pts]** We will now examine the utility of a line profiler to do per-line CPU profiling. This way we can actually evaluate exactly which line of code is utilizing our CPU resources.

You can install the profiler using : pip install line_profiler .

Documentation can be found here : https://github.com/rkern/line_profiler

You can now profile your code by decorating the appropriate function with "@profile" as described in the documentation. You can then run 'python -m line_profiler efficient_sparse_dot.py.lprof'. Please profile the dot_prod method in sparse_dot.py and efficient_sparse_dot.py, and report the

line of code which takes the maximum time, along with the amount of time taken by it.

2. **Memory Profiling**

   Next we will profile CPU Usage and memory usage using psutil. You can find the documentation of psutil here : https://psutil.readthedocs.io/en/latest/ . It is useful mainly for system monitoring, profiling, limiting process resources and the management of running processes. We will be looking at the memory_percent utility, which compares process memory to total physical system memory and calculates process memory utilization as a percentage.

   You can do this by first installing the psutil package. import psutil myProcess = psutil.Process(os.getpid())

   print(myProcess.memory_percent())

   You can also use psutil to check CPU utilization, by specifying print(psutil.cpu_percent()). Please profile sparse_dot.py and efficient_sparse_dot.py, and report the following:

   (a) **[1 pts]** Memory Percentage for sparse_dot.py:

   1.0789632797241 21

   (b) **[1 pts]** Memory Percentage for efficient_sparse_dot.py:

   2.7022600173950195

5 Submission [1 pts]

1. **[1 pts]** In addiiton to the PDF submission with your written answers and plots, please zip the files `bayes_net_empty.py` and `tree.py` directly (do NOT place them in a folder and then zip), name the zipped file `<andrewid>-10607-hwb.zip`, and submit the zipped folder on Gradescope under the assignment `Homework B (Programming)`. Your code file submissions should follow the provided code template files. In particular, they should NOT include any additional Python imports such as `matplotlib` which might make it fail on the Autograder. Please maintain any code for creating your plots in separate files. You do not have to submit your plotting code on Gradescope. Have you made the code submission on Gradescope? **Select one:**

   ● Yes

   ○ No

6 Collaboration Policy

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details including names of people who helped you and the exact nature of help you received.

   I received help from Zixuan Zhu for question2. 3(b)

2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details including names of people you helped and the exact nature of help you offered.

3. Did you find or come across code that implements any part of this assignment? If so, include full details including the source of the code and how you used it in the assignment.