The purpose of this first problem set is to remind you of the material you need from 2574 and help get you up-to-speed with python. This course uses these concepts from the first few lectures through the end of the semester, so it is important that you understand this material.

You can complete the exercises by either directly marking up this pdf, or by printing, completing, and scanning as a pdf. You should complete the Engineering Design Problems by writing the python code as instructed. The resulting pdf and python files should be uploaded to Canvas via the assignment tab by the due date and time.

Exercises

1. Determine the stack (LIFO queue) contents at the points indicated below during the following operations on a C++ stack. Write down the stack contents after the operation on the given line is executed. Be sure to indicate the top of the stack.

```
1. stack<int> s;
2. s.push(-1);
3. s.push(2);
4. s.push(-4);
5. s.push(200);
6. s.pop();
7. s.pop()
8. s.pop();
```

   (a) (2 points) After line 2:

   (b) (2 points) After line 5:

   (c) (2 points) After line 8:

2. Determine the FIFO queue contents at the points indicated below during the following operations on a C++ queue. Write down the queue contents after the operation on the given line is executed. Be sure to indicate the front of the queue.

```
1. queue<int> q;
2. q.push(-1);
3. q.push(2);
4. q.push(-4);
5. q.push(200);
6. q.pop();
7. q.pop()
8. q.pop();
```

(a) (2 points) After line 2:

(b) (2 points) After line 5:

(c) (2 points) After line 8:

3. Consider an array of values as follows. Rearrange them to form a (min) heap. Draw as both an array and as a tree.

   `{ 8, 12, -5, 32, -5, -7, 9, 84, -7, 5, 120 }`

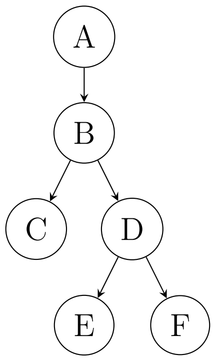
   (a) (3 points) As an array:

   (b) (3 points) As a tree:

4. (6 points) What is the average asymptotic complexity in big-O notation of the following data structures implementing a Dictionary ADT?

Operation	Hash Table	Balanced Binary Search Tree
insert		
remove		
search		

5. Consider the following binary tree



and assume children are expanded in left-right order. What order are the nodes visited in a

(a) (3 points) Preorder Traversal:

(b) (3 points) Inorder Traversal:

(c) (3 points) Postorder Traversal:

6. (5 points) Suppose I have an algorithm that uses a dictionary implemented as a Hash Table containing N items. The algorithm first searches for a given key/value pair and, if found, removes the pair, changes the value, and reinserts the new key/value. What is the average asymptotic complexity in big-O notation of this algorithm ?

Engineering Design Problems

To complete the engineering design problems, you will need to download the starter code in the ps0.zip file from the website.

7. (16 points) Implement a Python class called `Deque` in a file called `deque.py` that implements the following double-ended queue API:

```
Deque

class Deque
 |  A Deque class implementation.
 |
 |  Methods defined here:
 |
 |  __init__(self)
 |      Construct a Deque (initially empty)
 |
 |  back(self)
 |      return the back of the Deque
 |
 |  empty(self)
 |      return True is the Deque is empty, else False.
 |
 |  front(self)
 |      return the front of the Deque
 |
 |  pop_back(self)
 |      pop a value from the back of the Deque
 |
 |  pop_front(self)
 |      pop a value from the front of the Deque
 |
 |  push_back(self, value)
 |      push a value onto the back of the Deque
 |
 |  push_front(self, value)
 |      push a value onto the front of the Deque
```

Your implementation should pass the unit tests defined in `test_deque.py`. Submit only the `deque.py` file to Canvas.

8. (6 points) Consider a binary tree in Python represented as an adjacency list using a dictionary of tuples. For example the tree in exercise 5 could be written as

```
tree = {'A':('B',None), 'B':('C', 'D'), 'C': (None, None),
        'D':('E', 'F'), 'E':(None,None), 'F':(None, None)}
```

Write the python functions `preorder(tree, root)`, `inorder(tree, root)`, and `postorder(tree, root)`, defined in the file `tree.py`, to return a list of the tree nodes from the indicated traversal, starting at the subtree `root`.. Your implementation should pass the basic unit tests defined in `test_tree.py`. Submit only the `tree.py` file to Canvas.