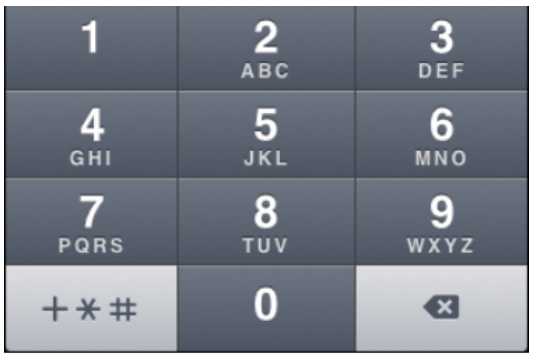
# CS5001 Fall 2021 Final Exam Review – Additional Questions

## 1. Data Structures

1.1



Given the above traditional alpha-numeric mapping, create a dictionary that corresponds to this mapping. Use the integer as the key and the letters as a single string such that if you execute the following  statement.

**print(phone_dict[8])**

The following will be printed to the console.
**"TUV".**

1.2
What two operations are required by a Stack Data Structure and what do they do?

What two operations are required by a Queue Data Structure and what do they do?

Describe a real-world scenario where you would implement a Stack and one where you would implement a Queue

## 2. Algorithms & Big OH

2.1
Sort the following Big-Oh time complexities from slowest to fastest.

O(n lg n)     O(5000)     O(2^n)     O(n)     O(n^2)     O(lg n)

2.2
Can you run binary search on the following array? A = [1, 7, 2, 5, 3, 6, 5]
Why or why not?

2.3
Briefly describe any sorting algorithm. What are its steps?
What is its running time? (Big Oh)

## 3. Classes

3.1
Create a Class Dog that has the following attributes: a name, a breed (both required as parameters in the constructor) and a boolean isFull which indicates whether it is full or hungry. By default isFull should be set to True.
The Dog Class should have two methods: (1) bark, which will print "woof"to the console and (2) eat, which will update isFull to False.

## 4. Recursion

4a. Create a recursive function that checks if a number is a power of 3.
Example: power_of_three(3) = True
power_of_three(1) = True
power_of_three(78) = False
power_of_three(0) = False
power_of_three(59049) = True

4b. Create a recursive implementation that flattens a nested list. (HARD)
Example: flatten_list([[1], [2, 3], [4], [3, [2, 4]]]) = [1, 2, 3, 4, 3, 2, 4]

## Solutions:

4a.

```
def power_of_three(number):
      if number == 1:
            return True
      elif number % 3 != 0 or number <= 0:
            return False
      else:
            return power_of_three(number // 3)
```

4b.

```
def flatten_list(lst):
      output = []
      for each in lst:
            if type(each) is list:
                  output.extend(flatten_list(each))
            else:
                  output.append(each)
      return output
```