CSC 404 - Activity/Project 8 - Name:

Problem 1.

a. Describe how `Mergesort` would sort this list (i.e., draw the nice graph to show how things come together).

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

b. Describe how `Quicksort` would sort this list (Use the 'middle' entry in the list(s) as your pivot)

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

c. Describe how `Quicksort` would sort this list (Use the first entry in the list(s) as your pivot)

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

d. Implement `Mergesort` and `Quicksort` in a language of your choice (be adventurous/have fun). Then test the above lists (as well as some other fun lists).

Problem 2 (Let's...find 4 - Weee!)**.** Suppose someone picks a number x from a set of n numbers. A second person tries to guess the number by successively selecting subsets of the n numbers and asking the first person whether x is in each set. The first person answers either 'yes' or 'no'. We can find x using $\log_2(n)$ queries ($\lceil \log_2(n) \rceil$ if n is not a power of 2) by successively splitting the sets used in each query in half.

The following algorithm, `findX`, returns the mystery value `x` of an input list `L`.

```
1.  if len(L) < 2, then return L
2.  else
        a.  L1 = first half of L
        b.  L2 = second half of L
        c.  If x in L1, then return findX(L1,x)
        d.  else return findX(L2,x)
3.  return L
```

a. Demonstrate this algorithm by finding the '4' in the following lists:

1	3	7	4	2	0	6	5

8	6	7	5	3	0	9	11	13	17	4	2	15	10	12	1

b. Implement this algorithm in a language of your choice and run it on the above lists (still looking for the mystery 4). I know, it is a really exciting function with the most exciting output - more about this in the next problem!

Problem 3. Ulam's problem ask for the number of queries required to find x, supposing that the first person is allowed to **lie** exactly once.

a. Option 1: [Ask each Question Twice!] Show that by asking each question twice, given a number x and a set with n elements, and asking one more question when we find the lie. Ulam's problem can be solved in at most $2\log_2(n) + 1$ queries. Demonstrate this process by finding the 4 in the following lists (you may choose when the 'lie' occurs)

1	3	7	4	2	0	6	5

8	6	7	5	3	0	9	11	13	17	4	2	15	10	12	1

b. (Bonus) Option 2: [Divide-and-Conquer!] Show that by dividing the initial set of n elements into four parts, each with $n/4$ elements, $1/4$ of the elements can be eliminated using two queries. [Hint: Use two queries, where each of the queries asks whether the element is in the union of two of the subsets with $n/4$ elements and where one of the subsets of $n/4$ elements is used in both queries.] Demonstrate this process by finding the 4 in the following lists (you may choose when the 'lie' occurs)

1	3	7	4	2	0	6	5

8	6	7	5	3	0	9	11	13	17	4	2	15	10	12	1

c. (Bonus) Is the naive way to solve Ulam's problem by asking each question twice or the divide-and-conquer method more efficient?