CSCI 5454 Spring 2023: Make up assignment for exam # 1
Due Date: Friday, April 7th

Important Note: Your adjusted grades will not show up on canvas. They will be calculated when we compute your final grade at the end of the semester. **Please do not keep asking us about it on piazza or email.** The exact grade adjustment formula is as follows:

$$\max(E, 0.6E + 0.4M)$$

where E is original exam grade and M is make up exam grade. Note that if you miss this assignment, it will keep your exam grade unchanged.

P1. Let n be a number divisible by 3. Write down the FFT of the sequence:

$$\underbrace{1, 0, 0, 1, 0, 0, \ldots, 1, 0, 0}_{\text{length}=n}$$

Write down the FFT coefficients and show your calculations justifying your answer.

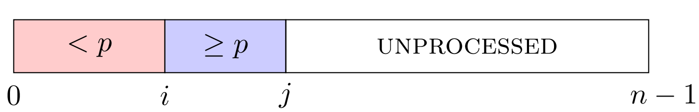
P2. A partition of an array A of n integers w.r.t to a pivot p separates the array into two parts: the left partition has all elements $< p$ and the right partition has all the elements $\geq p$.
Example: Consider the example below:

Inputs: Array A $[1, 5, 2, -2, 3, 4, 8, 9, 0]$, and pivot $p = 5$.

Output: Array modified in-place to $[1, 0, 2, -2, 3, 4, \underline{5}, \underline{9}, \underline{8}]$ and return index $k = 6$ to indicate the position where the right partition begins.

Note: There is no requirement on the order of the elements in each partition. All we require is that the left partition $A[0], \ldots, A[k-1]$ contains all elements $< p$ and right partition $A[k], \ldots, A[n-1]$ contains all elements $\geq p$.

Consider the following algorithm for partitioning. At any step, it maintains two indices into the array i, j and divides the array into three parts:



Loop Invariant

- All elements $A[0], \ldots, A[i]$ (inclusive) are guaranteed $< p$.
- All elements $A[i+1], \ldots, A[j]$ (inclusive) are guaranteed $\geq p$.
- Elements from $A[j+1], \ldots, A[n-1]$ do not have any guarantees (unprocessed).

Complete the missing portions of the code below.

```
def partition(A, p):

    i =                        # todo: initial value of i


    j =                        # todo: initial value of  j

    while (j < n-1):           # while some unprocessed element remains...
        if (A[j+1] < p):       # element A[j+1] must be belong to "first" partition
            swap(A, i+1, j+1)  # Swap A[i+1] and A[j+1]
            i = i + 1          # First partition now goes up to i + 1
            j = j + 1          # Second partition now goes up to j + 1
        else:                  # element A[j+1] must belong to the "second" partition
                               # todo: fill out missing code to update i, j for else branch.



    return                     # todo: what index should we return corresponding to k?
```

What is the running time of the algorithm above in terms of n?