CMPS 101 HW Assignment

1. Pseudo code

```
i=1
for i in (length of A)-1
    temp=i, k=i+1
    for k in range A[i] to length of A
        if A[k] < A[temp]
            temp = k
    holder = A[i]
    A[i] = A[temp]
    A[temp] = holder
```

The loop invariant is checking wether the indicy of the smallest value is smaller then the indicy of the compared value. It will only need to run n-1 times because the last element, n, will be the largest element, so no sorting is needed at the nth time.

best Θ	$\Theta(n^2)$
avg Θ	$\Theta(n^2)$
worst Θ	$\Theta(n^2)$

2. Preform merge sort on array A

A = (3, 41, 52, 26, 38, 57, 9, 49)

(3, 9, 26, 38, 41, 49, 52, 57)

merge

(3, 26, 41, 52) (9, 38, 49, 57)

merge

(3, 41) (26, 52) (38, 57) (9, 49)

merge

(3) (41) (52) (26) (38) (57) (9) (49)

3.

$$T(n) = \begin{cases} g & \text{if } n = 1 \\ T(n-1) \cdot g(n-1) & \text{if } n > 1 \end{cases}$$

4. Pseudocode

```
BinarySearch(A, p, r, v)

if p≤r{
   mid = ⌊((r-q)+1)/2⌋
 if A[mid] > v
   BinarySearch(A, p, mid-1, v)
 if A[mid] < v
   BinarySearch(A, mid+1, r, v)
 if A[mid] == v   return mid
else:
   return -1  (failed)
```

The worst case for binary search is $\Theta(\lg n)$ which makes sense because in each recurrence you reduce the size of the array by half eventually to reach only 1 value.
$\therefore$ log2 of n gives the maximum number of times an array will be halved before 1 value is found.

5. a. (2,5), (3,4), (3,5), (4,5), (1,5)
b. The array [n, n-1, . . . , 3, 2, 1] will have most inversions, it will have $\sum_{i=1}^{n} n-i$ inversions.
c. The relationship between the number of inversions and running time is that n-1 comparisons are preformed n times inversions are the number of times the the comparison for inversion was satisfied as true. True and flase statemenst both count in run time so $n \cdot n-1$ comparision makes a worst case of $\Theta(n^2)$.
d. Inversion(A, p, r)

```
mid = ⌊(p+r)/2⌋
Inversion(A, p, mid)
Inversion(A, mid+1, r)
i = p
while (i ≤ mid)
    if mid+1+i > r
        break
    else:
        if A[i] > A[mid+1+i]
            print(i, mid+1+i)
    i++
```