CSI2110 - Assignment 1

Brennan McDonald - 8195614

1. a) Best case - O(n log n)
   b) Worst case - O(n^2)

2. a)

$7n^3 + 3n^2 - 2n + 100 \leq Cn^3$

$3n^2 \leq 3n^3 \text{ for all } n \geq 1$

$2n \leq 2n^3 \text{ for all } n \geq 1$

$100 \leq 100n^3 \text{ for all } n \geq 1$

$7n^3 + 3n^3 - 2n^3 + 100n^3 = 108n^3$

$c = 108,\ n_0 = 1$

b)

$\frac{(n^2+1)}{(n+1)} \leq Cn$

$\text{for } n \geq 1,\ C = 1 \text{ because n is always greater than } \frac{(n^2+1)}{(n+1)}$

c) $n! \leq cn^n \text{ for } C = 1 \text{ when } n \geq n_0 = 1$

3. a)

```
def alg(A):
        a := 0
        b := 0
        a1 := 0
        b1 := 0
        For each element e in A:
                if the index of e = 0:
                        a := 0
                else:
                        if the previous element is greater than e:
                                If b-a > b1 - a1:
                                        a1 := a
                                        b1 := b
                                a := index of e
                        else :
                                b := index of e
        return a1 and b1
```

   b) The big Oh of my algorithm in O(n) because there is only one iteration of the elements

4. a)

```
def alg(n):
        A := the numbers from 1 to n+1 with one number missing
        pivot := n/2
        while true:
                if pivot = A[pivot] and pivot+1 < A[pivot+1]:
                        return pivot + 1
                if pivot = A[pivot]:
                        pivot := pivot + ((pivot+1)/2)
                else if pivot < A[pivot]:
                        pivot := floor((pivot+1)/2)
```

b) Since the algorithm uses a heuristic to find out which element to hit next and that heuristic involves dividing the searched section by 2 the

5. Operation Output Q

enqueue (4) -> [4]
dequeue () -> []
dequeue () -> []
enqueue (44) -> [44]
enqueue (7) -> [44,7]
enqueue (6) -> [44,7,6]
dequeue () -> [7,6]
isEmpty() -> false
enqueue (3) -> [7,6,3]
enqueue(5) -> [7,6,3,5]
dequeue () -> [6,3,5]
dequeue () -> [3,5]
dequeue () -> [5]
dequeue () -> []
enqueue(32) -> [32]
enqueue(39) -> [32,39]
enqueue(9) -> [32,39,9]
size() -> 3
enqueue(32) -> [32,39,9,32]
size() -> 4
dequeue() -> [39,9,32]
enqueue(6) -> [39,9,32,6]
enqueue (5) -> [39,9,32,6,5]
Dequeue() -> [9,32,6,5]
front() -> 9
size() -> 4
enqueue (9) -> [9,32,6,5,9]

6. $x^2 * sin(x) + x^2$

7. It would have a big-Oh characterization of n^4 because the first loop would run from 0 to n*n and the inner loop would also run from 0 to n*n, this would cause an n^4 complexity.