HW 5 - Cougar Bellinger

1)

```
INPUT: A array of N integers where N >= 3
OUTPUT: Three largest elements in array A

first = 0
second = 0
third = 0

for i = 0 to N-1:
    if A[i] > first:
        third = second
        second = first
        first = A[i]
    else if A[i] > second:
        third = second
        second = A[i]
    else if A[i] > third:
        third = A[i]

return [first, second, third]
```

2)

The average case for a linear search is **n/2**. Assuming the search value is at iteration i, it takes the linear search i iterations to find the search value. Since the average case for the search key is to be at n/2, that means it would take n/2 iterations on average.

3)

The average case for a binary search is **log(n)** because it takes $\log_2(n)$ steps to reduce n to 1 by dividing by 2, and we always use these halving operations regardless of the target's position.

4)

Linear search has the advantage over binary search in the case of when the array is **unsorted**. The linear search starts at the first element, regardless of the order of the array, while binary search relies on the properties of sorted arrays.

5)

Advantages of Hashtables:

1. **Fast Access Time**: Provides O(1) average case time complexity for insertion, deletion, and search operations
2. **Memory Efficient**: Only stores the data that is actually needed, with no requirement for pre-allocated space like arrays
3. **Dynamic Size**: Can grow or shrink based on the amount of data being stored, making them flexible

Disadvantages of Hashtables:

1. **Collision Handling**: When two keys hash to the same location, collisions must be handled using methods like chaining or open addressing
2. **No Ordering**: Elements are not stored in any particular order, making it inefficient for operations that require sorted data
3. **Space Overhead**: Additional memory is required for storing the hash table structure and handling collisions

6)

Advantages of Search Trees:

1. **Ordered Data Structure**: Maintains elements in sorted order, making operations that depend on sorted data efficient
2. **Balanced Performance**: Provides O(log n) time for search, insert, and delete operations
3. **Flexible Size**: Can dynamically grow and shrink as needed

Disadvantages of Search Trees:

1. **Balance Maintenance**: Extra overhead required to maintain balance in the tree, which complicates implementation
2. **Space Usage**: Each node requires extra space for pointers to child nodes, making them less memory efficient
3. **Complex Implementation**: Can be difficult to implement