**Lab W1D3**

**Question 1**

Func beautiful(A, i)

return A[i]

Direct access of array elements: Accessing an element by index in an array takes constant time O (1) regardless of the element's position or value.

**Question 2**

1. 2 ^ n
2. 2 ^ (n + 1) = 2 \* (2 ^ n)
3. 2 ^ (2n) = (2 ^ 2) ^ n = 4 ^ n
4. 2 ^ ( 2 ^ n )

**Question 3**

1. O (1): direct access to element in an array at index.
2. O (log n): binary search.
3. O (n): linear search.
4. O (n log n): merge sort.
5. O (n ^ 2): insertion sort.
6. O (n ^ 3): Floyd-Warshall Algorithm.
7. O (2 ^ n): Naive Recursive Fibonacci.

**Question 4**

We cannot apply the Master Theorem to determine the time complexity of the naive recursive Fibonacci algorithm because its recurrence relation does not fit the form required for the Master Theorem.

The naive Fibonacci recurrence is:

T(n) = T(n – 1) + T(n-2) + O(1)