Question 1

(a) memo = empty dictionary

```
function Fibonacci (n, memo):
    if n in memo:
        return memo(n)
    if n == 0:
        return 0
    if n == 1:
        return 1
    memo[n] = Fibonacci (n-1, memo) + Fibonacci (n-2, memo)
    return memo[n]
```

(b) $O(n)$

Question 2

(a) Let $LIS(i)$ be the length of the longest increasing subsequence that end at index i

(b) To compute $LIS(i)$, check all previous indices j where $j < i$ and $A[j] < A[i]$. choose the max $LIS(j)$ among all valid j and extend it by including $A(i)$

(c) $LIS(i) = 1 + \max\{LIS(j) \mid 0 \le j \le i \text{ and } A(j) < A(i)\}$

If no such j exists, then $LIS(i) = 1$

(d) $LIS(0) = 1$, which means there the LIS end at first element.

(e) Total number of subproblems is n, And running time for each subproblem is $O(n)$. because every subproblems need to check all previous $j < i$.

Total number of subproblems is n

running time for each subproblem is $O(n)$

(f) Init: $dp = [1, 1, 1, 1, 1, 1]$

And we need to compute $dp[i]$ for $i = 1$ to 5

$i = 1$, $A[1] = 14$,

$A[0] = 11 < 14$, $dp[1] = \max(dp(1), dp(0) + 1) = 2$

$dp = [1, 2, 1, 1, 1, 1]$

$i = 2$, $A[2] = 13$

$j = 0$, $A[0] = 11 < 13$, $dp(2) = \max(1, 1+1) = 2$

$j=1$, $A[1]=14>13$, skip

$dp=[1,2,2,1,1,1]$

$i=3$, $A[3]=7$

$A[2]=13>7$, skip

$i=4$. $A[4]=8$,

$A[3]=7<8$, $dp[4]=\max(dp(4), dp(3)+1)=2$

$dp=[1,2,2,1,2,1]$

$i=5$. $A[5]=15$,

$A[4]=8<15$, $dp[5]=\max(dp(5), dp(4)+1)=3$

$dp=[1,2,2,1,2,3]$

so $LIS=3$