1. 
   a) function Compute (n) 
      
      \[
      \text{sum} = 0 \\
      \text{if } (n=0) \text{ or } (n=1) \text{ then} \\
      \quad \text{return 1} \\
      \text{else} \\
      \quad \text{for } i=1 \text{ to } n-1 \text{ do} \\
      \quad \quad \text{sum} = \text{sum} + \text{Compute} (i) \times \text{Compute} (i-1) + 1 \\
      \quad \text{endfor} \\
      \quad \text{sum} = \text{sum} + \text{Compute} (n-1) \\
      \text{return sum} \\
      \text{endif} \\
      \end
      
      Solution of \( T(n) \): 
      
      \[
      T(n) = T(0) + 2[T(1) + T(2) + \cdots + T(n-2) + T(n-1)] + (n - 1)
      \]
      
      So, 
      
      \[
      T(n - 1) = T(0) + 2[T(1) + T(2) + \cdots + T(n-3) + T(n-2)] + (n - 2)
      \]
      
      Subtracting the second one from the first, we obtain 
      
      \[
      T(n) = 3T(n-1) + 1
      \]
      
      Solving by backward substitution, 
      
      \[
      T(n) = 3^{n-1} + \sum_{i=0}^{n-2} 3^i = 3^{n-1} + \left( \frac{3^{n-1} - 1}{2} \right) \in \Theta(3^n)
      \]
      
   b) function Compute (n) 
      
      \[
      T[0] = 1 \\
      T[1] = 1 \\
      \text{for } i=2 \text{ to } n \text{ do} \\
      \quad T[i] = 0 \\
      \quad \text{for } j=1 \text{ to } i-1 \text{ do} \\
      \quad \quad T[i] = T[i] + T[j] \times T[j-1] + 1 \\
      \quad \text{endfor} \\
      \quad T[i] = T[i] + T[i-1] \\
      \text{endif} \\
      \text{return } T[n] \\
      \end
      
      \[
      T(n) = \sum_{i=2}^{n} \left( \sum_{j=1}^{i-1} 1 \right) + 1
      \]
      
      \[
      T(n) = \sum_{i=2}^{n} i = \frac{n(n-1)}{2} - 1 \in \Theta(n^2)
      \]
c) function Compute\( (n)\)
\[
\begin{align*}
T[0] & = 1 \\
T[1] & = 1 \\
T[2] & = T[0] \times T[1] + 2 \\
\text{for } i=3 \text{ to } n \text{ do } \\
& \quad T[i] = T[i-1] + (T[i-1] \times T[i-2] + 1) - T[i-2] + T[i-1] \\
\text{endfor} \\
\text{return } T[n] \\
\end{align*}
\]

\[T(n) = \sum_{i=3}^{n} 1 \in \Theta(n)\]

2. Theorem: Given integers \(n, k, \) and \(L[1:n]\), suppose \(L[1:n]\) is a list such that every element in the list is no more than \(k\) positions from its stable final position in the sorted list \(L\). Then insertion sort performs at most \(2k(n-1)\) comparisons when sorting \(L[1:n]\).

First, we will show that, if each element in the list is no more than \(k\) positions from its stable final position, then for each \(i \in \{2,\ldots,n\}\), there are at most \(2k-1\) list elements \(L[j]\) such that \(j<i\) and \(L[i]<L[j]\).

Assume to the contrary that there are at least \(2k\) list elements such that \(j<i\) and \(L[i]<L[j]\). Then there must exist a list element \(L[j_0]\) that is strictly greater than \(L[i]\), such that \(j_0 \leq 2k\). Let \(i'\) and \(j_0'\) denote the stable final positions of \(L[i]\) and \(L[j_0]\), respectively. By hypothesis, every element in the list \(L[1:n]\) is no more than \(k\) positions from its stable final position. In particular, \(j_0' \leq j_0 + k \leq (i-2k)+k=i-k\), and \(i' \leq i-k\). Hence, \(j_0' \leq i'\), which implies that \(L[j_0] \leq L[i]\), a contradiction.

From the conclusion that there are at most \(2k-1\) list elements \(L[j]\) such that \(j<i\) and \(L[i]<L[j]\) and the fact that the algorithm iterates \(n-1\) times, the theorem follows.

3.
\begin{tabular}{|c|c|c|}
\hline
a) & Visit & Unvisited neighbors \hspace{1cm} Backtrack \\
\hline
1 & 5,6,7,8,9 & \\
5 & --- & to 1 \\
1 (returned) & 6,7,8,9 & \\
6 & 3,4,8 & \\
3 & 4,7 & \\
4 & 8 & \\
8 & 9 & \\
9 & 2 & \\
2 & 10 & \\
10 & --- & to 2 \\
2 (returned) & --- & to 9 \\
9 (returned) & --- & to 8 \\
8 (returned) & --- & to 4 \\
4 (returned) & --- & to 3 \\
\hline
\end{tabular}
So, order of visits: 1, 5, 6, 3, 4, 8, 9, 2, 10, 7

b) Visit   Unvisited neighbors   Enqueue
1   5, 6, 7, 8, 9
5, 6, 7, 8, 9
5 (dequeue)   ---
6 (dequeue)   3, 4
7 (dequeue)   ---
8 (dequeue)   ---
9 (dequeue)   2
3 (dequeue)   ---
4 (dequeue)   ---
2 (dequeue)   10
10 (dequeue)   ---

So, order of visits: 1, 5, 6, 7, 8, 9, 3, 4, 2, 10

4. We can view the algorithm as having two steps. Let $T_1$ denote the number of basic operations in the loop and $T_2$ the number of basic operations in the recursive calls. Then

$$A(n) = E[T] = E[T_1] + E[T_2]$$

Similarly, we can divide the work inside the loop into two parts: Let $T_{1,1}$ be the number of times first basic operation is executed and $T_{1,2}$ the number of times second basic operation is executed. Then

$$E[T_1] = E[T_{1,1}] + E[T_{1,2}] = (n-1) + E[T_{1,2}]$$

We can assume that it is equally likely that $L[\text{low}]$ can be any one of the integers 1,...,n. So, the second print(..) statement will be executed (n-1) times with probability 1/n, will be executed (n-2) times with probability 1/n, ..., will be executed 0 times with probability 1/n. Thus

$$E[T_{1,2}] = \sum_{i=0}^{n-1} i \cdot \frac{1}{n} = \frac{1}{n} \cdot \frac{n(n-1)}{2} = \frac{n-1}{2}$$

Then, $E[T_1] = \frac{3(n-1)}{2}$. Then, assuming that the random(..) command returns any number between 1 and n with equal probability,

$$A(n) = \frac{3(n-1)}{2} + \frac{1}{n} \sum_{i=1}^{n} A(i) + A(n - i + 1) \cdot A(1)=0$$
\[ A(n) = \frac{3(n - 1)}{2} + \frac{2}{n}[A(1) + \cdots + A(n)] \]

Multiply with \( n \):

\[ n \ (n) = \frac{3n(n - 1)}{2} + 2[A(1) + \cdots + A(n)] \]

Replace \( n \) with \( n - 1 \):

\[ (n - 1)A(n - 1) = \frac{3(n - 1)(n - 2)}{2} + 2[A(1) + \cdots + A(n - 1)] \]

Subtract the second one from the first:

\[ n \ (n) - (n - 1)A(n - 1) = \frac{6(n - 1)}{2} + 2A(n). \] Then

\[ (n - 2)A(n) = (n - 1)A(n - 1) + \frac{6(n - 1)}{2} + 2A(n). \]

Divide both sides to \((n - 1)(n - 2)\):

\[ \frac{A(n)}{n - 1} = \frac{A(n - 1)}{n - 2} + \frac{3}{2(n - 2)}. \]

Let \( y(n) = \frac{A(n)}{n - 1} \). Then

\[ y(n) = y(n - 1) + \frac{6}{2(n - 2)}, \quad y(1) = 0 \]

When we solve \( y(n) \) with backward substitution, we will obtain

\[ y(n) = 3 \sum_{i=1}^{n-2} 1 \cong H(n). \] Thus,

\[ A(n) \cong (n - 1)H(n) \in \Theta(n). \]