Take home activities

Activity 1:

Little Red Riding Hood was a kind girl who always wore a red hooded cape. One day, her mother asked her to take a basket of treats to her sick grandmother, reminding her to stay on the path and not talk to strangers.

On her way, she met a sly wolf who tricked her into telling him where her grandmother lived. The wolf ran ahead to the grandmother's house, locked her in a closet, and disguised himself in her clothes.

When Little Red Riding Hood arrived, she noticed something strange about her "grandmother." "What big eyes you have!" she said. "What big teeth you have!" The wolf replied, "All the better to eat you with!" and lunged at her.
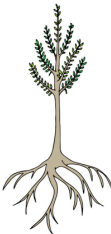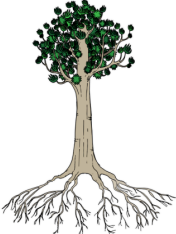
Just then, a woodsman nearby heard her scream, rushed in, and chased the wolf away, saving Little Red Riding Hood and her grandmother. Little Red Riding Hood learned to always listen to her mother's advice and stay safe.

Sort the images below to match the story:

1.	2.	3.

5.	4.	6.

Activity 2 Kauri Tree

Sort the images below in the correct order:

1.	2.	3.
5.	4.	6.

Sorting algorithms

Selection Sort: repeatedly finding the minimum element from the unsorted part of the array and swapping it with the first unsorted element.

Example: [29, 10, 14, 37, 13]

Step-by-Step Solution:
Initial List: [29, 10, 14, 37, 13]
- Pass 1:
  - Find the minimum in the list [29, 10, 14, 37, 13], which is 10.
  - Swap 10 with the first element (29).
  - Result: [10, 29, 14, 37, 13]

- Pass 2:
  - Find the minimum in the remaining list [29, 14, 37, 13], which is 13.
  - Swap 13 with the second element (29).
  - Result: [10, 13, 14, 37, 29]
- Pass 3:
  - Find the minimum in the remaining list [14, 37, 29], which is 14.
  - Swap 14 with itself (no change).
  - Result: [10, 13, 14, 37, 29]

- Pass 4:
  - Find the minimum in the remaining list [37, 29], which is 29.
  - Swap 29 with 37.
  - Result: [10, 13, 14, 29, 37]

Final Sorted List: [10, 13, 14, 29, 37]

Bubble sort: sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted.

Example: [5, 3, 8, 4, 2]

   Step-by-Step Solution:
     Initial List: [5, 3, 8, 4, 2]
   ● Pass 1:
        ○ Compare 5 and 3: Swap, since 5 > 3. → [3, 5, 8, 4, 2]
        ○ Compare 5 and 8: No Swap, since 5 < 8. → [3, 5, 8, 4, 2]
        ○ Compare 8 and 4: Swap, since 8 > 4. → [3, 5, 4, 8, 2]
        ○ Compare 8 and 2: Swap, since 8 > 2. → [3, 5, 4, 2, 8]
   ● Pass 2:
        ○ Compare 3 and 5: No Swap. → [3, 5, 4, 2, 8]
        ○ Compare 5 and 4: Swap, since 5 > 4. → [3, 4, 5, 2, 8]
        ○ Compare 5 and 2: Swap, since 5 > 2. → [3, 4, 2, 5, 8]
        ○ Compare 5 and 8: No Swap. → [3, 4, 2, 5, 8]
   ● Pass 3:
        ○ Compare 3 and 4: No Swap. → [3, 4, 2, 5, 8]
        ○ Compare 4 and 2: Swap, since 4 > 2. → [3, 2, 4, 5, 8]
        ○ Compare 4 and 5: No Swap. → [3, 2, 4, 5, 8]
        ○ Compare 5 and 8: No Swap. → [3, 2, 4, 5, 8]
   ● Pass 4:
        ○ Compare 3 and 2: Swap, since 3 > 2. → [2, 3, 4, 5, 8]
        ○ Compare 3 and 4: No Swap. → [2, 3, 4, 5, 8]
        ○ Compare 4 and 5: No Swap. → [2, 3, 4, 5, 8]
        ○ Compare 5 and 8: No Swap. → [2, 3, 4, 5, 8]

Final Sorted List: [2, 3, 4, 5, 8]

Insertion Sort: sorting algorithm that builds the final sorted array one item at a time by comparing each new item to the already sorted items and inserting it into its correct position.

Example: [7, 4, 5, 2]

Step-by-Step Solution:
  Initial List: [7, 4, 5, 2]
- Pass 1:
    - Take the second element (4) and compare it with the first (7).
    - Since 4 < 7, insert 4 before 7.
    - Result: [4, 7, 5, 2]

- Pass 2:
    - Take the third element (5) and compare with the elements before it.
    - Compare 5 with 7: 5 < 7, so shift 7 to the right.
    - Insert 5 before 7.
    - Result: [4, 5, 7, 2]

- Pass 3:
    - Take the fourth element (2) and compare with the elements before it.
    - Compare 2 with 7: 2 < 7, so shift 7 to the right.
    - Compare 2 with 5: 2 < 5, so shift 5 to the right.
    - Compare 2 with 4: 2 < 4, so shift 4 to the right.
    - Insert 2 before 4.
    - Result: [2, 4, 5, 7]

Final Sorted List: [2, 4, 5, 7]

Merge Sort: a divide-and-conquer algorithm that divides the list into two halves, recursively sorts each half, and then merges the two sorted halves back together.

`Example: [9, 3, 7, 5]

Step-by-Step Solution:
Initial List: [9, 3, 7, 5]
- Divide: Split the list into two halves.
    - Left Half: [9, 3]
    - Right Half: [7, 5]

- Recursively Sort:
    - Sort Left Half [9, 3]:
        - Split into [9] and [3].
        - Merge: [3, 9]

    - Sort Right Half [7, 5]:
        - Split into [7] and [5].
        - Merge: [5, 7]
- Merge:
    - Merge [3, 9] and [5, 7]
        - Compare and merge in sorted order: [3, 5, 7, 9]

Final Sorted List: [3, 5, 7, 9]

Activity 3

1. Sort the following list in ascending order using the Selection Sort algorithm:
   [45, 22, 15, 8, 30]

2. Sort the following list in ascending order using the Bubble Sort algorithm:
   [5, 1, 12, -5, 16]

3. Sort the following list in ascending order using the Insertion Sort algorithm:
   [6, 4, 2, 8, 3]

4. Sort the following list in ascending order using the Merge Sort algorithm:
   [10, 7, 3, 8, 6, 2, 5, 4]