Assessment 2

Part A: Managing Arrays

1. Creating an unsorted array 'test' of 'size 5'

```
test = [29,28,12,21.3]
```

2. Increasing the test array size to 10 elements and naming it 'test1'

```
test1 = test + [4, 5, 6, 2, 8]
```

3. Finding the length of 'test1' array

```
length_test1 = len(test1)
```

4. Printing the contents of 'test1' array

```
print(f"The contents of test1 array are {test1}")
print(f"The length of test1 array is {length_test1}")
```

Part B: Creating an Abstract Data Type (ADT) for 'test1' array

> ADT Array:

```
  Operations:
    Create(size)
    Add(element)
    Remove(element)
    GetElement(index)
```

EndADT

Part C: Creating Pseudo-code for the sorting of 'test1' array

Bubble sort

> For I = 0 to length(test1) - 1
> For J = 0 to length(test1) - (i+1)
> IF test1[J] > test1[J+1]
> Swap test1[J], test1[J+1]
> End J
> End I

Selection Sort

> For I = 0 to length(test1) - 1
> For J = I+1 to length(test1) -1
> If test1[J] < test1[I]
> Swap test1[J], test1[I]
> End J
> End I

Part D: Python coding for bubble sort and selection sort

Bubble sort

```
def bubble_sort(arr):
        i_range = len(arr)
        for i in range(i_range):
                j_range = i_range-i-1
                for j in range(j_range):
                        if arr[j]>arr[j+1]:
                                arr[j], arr[j+1] = arr[j+1], arr[j]
        return arr
```

Selection sort

```
def selection_sort(arr):
        i_range = len(arr)
        for i in range(i_range):
                j_range = i_range - 1
                for j in range(i+1, j_range):
                        if arr[j] &lt arr[i]:
                                arr[j], arr[i] = arr[i], arr[j]
        return arr
```

Part E: Analysis of Time Complexity

- Bubble sort: The algorithm of bubble sort has a time complexity of O(n^2) in worst and average cases, O(n) in the best case which is when the array are already sorted.
- Selective sort: The algorithm of selective sort has a time complexity of O(n^2) in all cases.

Part F: Recommendation

Based on the performance analysis, while both algorithms exhibit a time complexity of O(n^2) for the average case, bubble sort may be slightly more efficient in nearly sorted lists due to its best-case scenario. However, selection sort consistently performs fewer swaps, which might be beneficial in scenarios where write operations are more costly.