4. Write programs in **any programming language** to implement the following sorting algorithms:

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**

In each implementation:

- Maintain a **counter variable** to record the **number of iterations** executed by:
    - the **outer loop**, and
    - the **inner loop**.
- Using this counter, determine and **compare the number of iterations** required for:
    - **Best Case**
    - **Average Case**
    - **Worst Case**

For each sorting algorithm, you must:

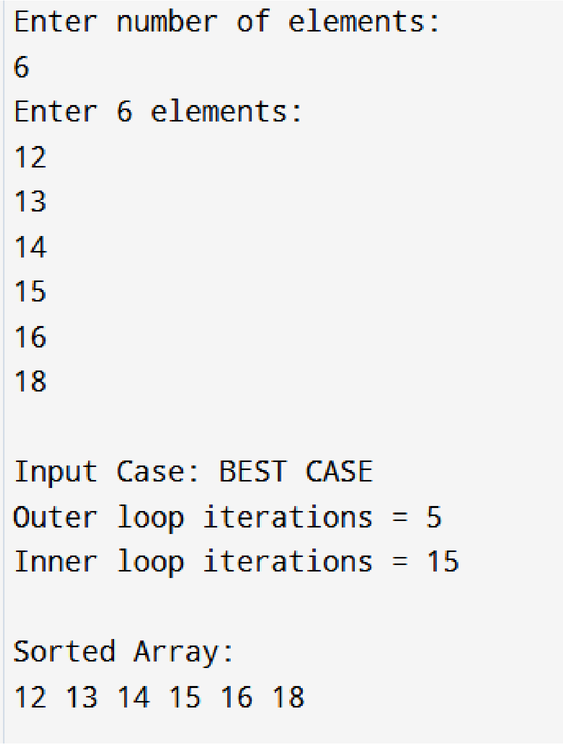
1. Clearly specify the **input used for best, average, and worst cases**.
2. Report the **iteration counts** for each case.
3. Analyze and compare the **time complexity behavior** based on the observed counts.
4. Present your results in a **table format** and give a brief conclusion.

Bubble Sort

Code:

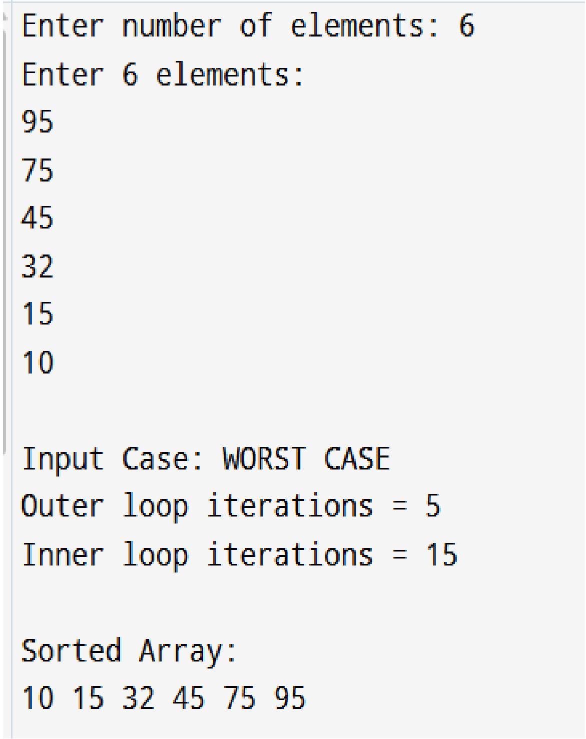
```
#include <stdio.h>
int isSortedAscending(int a[], int n)
{
    int i;
    for ( i = 0; i < n - 1; i++) {
        if (a[i] > a[i + 1])
            return 0;
    }
    return 1;
}

int isSortedDescending(int a[], int n) {
                int i;
    for ( i = 0; i < n - 1; i++) {
        if (a[i] < a[i + 1])
            return 0;
    }
    return 1;
}

void bubbleSort(int a[], int n) {
    int i, j, temp;
    int outer = 0, inner = 0;

    for (i = 0; i < n - 1; i++) {
        outer++;
        for (j = 0; j < n - i - 1; j++) {
            inner++;
            if (a[j] > a[j + 1]) {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }

    printf("\nOuter loop iterations = %d", outer);
    printf("\nInner loop iterations = %d\n", inner);
}

int main() {
    int n,i;
    int a[50];

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    if (isSortedAscending(a, n)) {
        printf("\nInput Case: BEST CASE");
    }
    else if (isSortedDescending(a, n)) {
        printf("\nInput Case: WORST CASE");
    }
    else {
        printf("\nInput Case: AVERAGE CASE");
    }
    bubbleSort(a, n);

    printf("\nSorted Array:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }

    return 0;
}
```

Best Case Output:



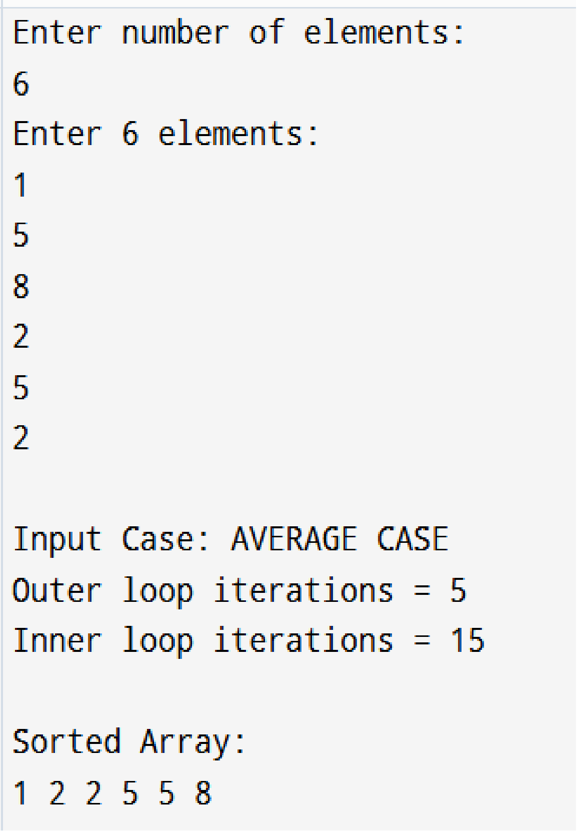
```
Enter number of elements:
6
Enter 6 elements:
12
13
14
15
16
18

Input Case: BEST CASE
Outer loop iterations = 5
Inner loop iterations = 15

Sorted Array:
12 13 14 15 16 18
```

Worst Case Output:



```
Enter number of elements: 6
Enter 6 elements:
95
75
45
32
15
10

Input Case: WORST CASE
Outer loop iterations = 5
Inner loop iterations = 15

Sorted Array:
10 15 32 45 75 95
```

Average Case Output:



```
Enter number of elements:
6
Enter 6 elements:
1
5
8
2
5
2

Input Case: AVERAGE CASE
Outer loop iterations = 5
Inner loop iterations = 15

Sorted Array:
1 2 2 5 5 8
```

Time Complexity:

- Best Case: $O(n^2)$
- Average Case: $O(n^2)$
- Worst Case: $O(n^2)$

Iteration Count:

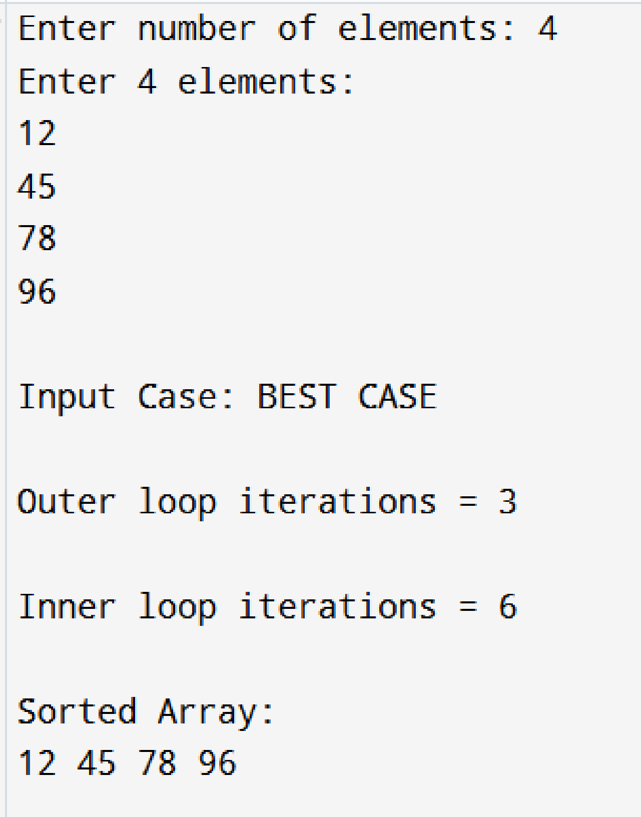
Case	Outer	Inner
Best	4	10
Average	4	10
Worst	4	10

Selection Sort:

```
#include <stdio.h>
int isSortedAscending(int a[], int n) {
        int i;
  for (i = 0; i < n - 1; i++) {
    if (a[i] > a[i + 1])
      return 0;
  }
  return 1;
}

int isSortedDescending(int a[], int n) {
        int i;
  for ( i = 0; i < n - 1; i++) {
    if (a[i] < a[i + 1])
      return 0;
  }
  return 1;
}
```

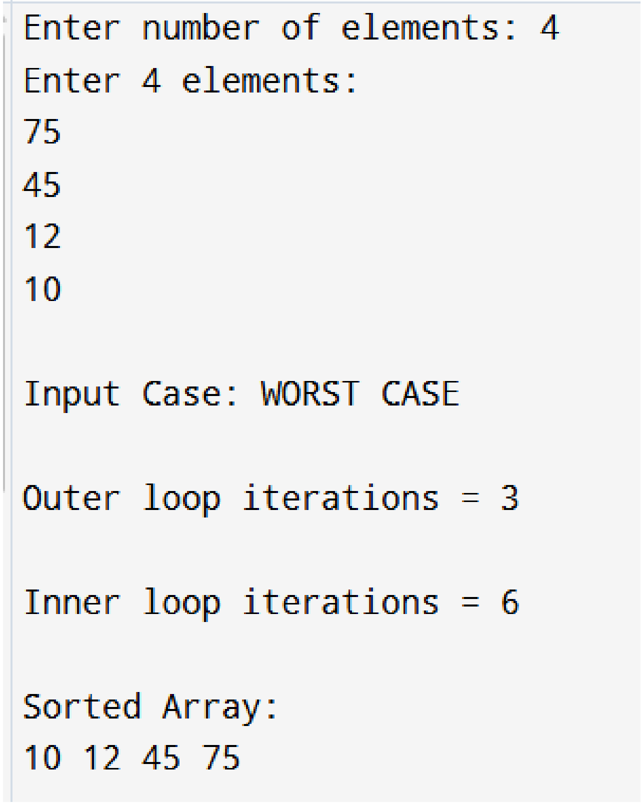
```
void selectionSort(int a[], int n) {
    int i, j, min, temp;
    int outer = 0, inner = 0;

    for (i = 0; i < n - 1; i++) {
        outer++;
        min = i;
        for (j = i + 1; j < n; j++) {
            inner++;
            if (a[j] < a[min]) {
                min = j;
            }
        }
        temp = a[i];
        a[i] = a[min];
        a[min] = temp;
    }

    printf("\nOuter loop iterations = %d\n", outer);
    printf("\nInner loop iterations = %d\n", inner);
}

int main() {
    int n,i;
    int a[50];

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    if (isSortedAscending(a, n))
        printf("\nInput Case: BEST CASE\n");
    else if (isSortedDescending(a, n))
        printf("\nInput Case: WORST CASE\n");
    else
        printf("\nInput Case: AVERAGE CASE\n");

    selectionSort(a, n);

    printf("\nSorted Array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);

    return 0;
}
```

Best Case Output:



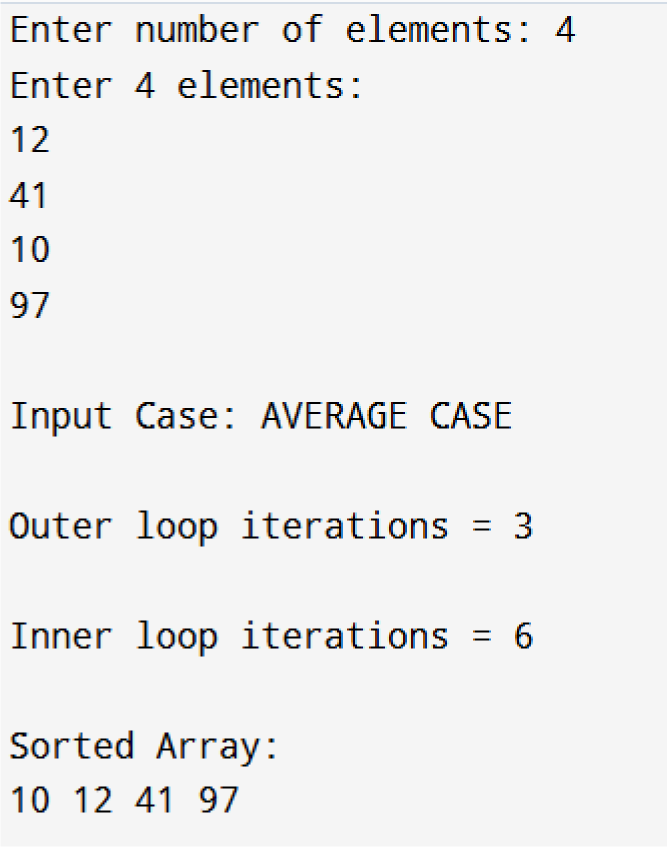
```
Enter number of elements: 4
Enter 4 elements:
12
45
78
96

Input Case: BEST CASE

Outer loop iterations = 3

Inner loop iterations = 6

Sorted Array:
12 45 78 96
```

Worst Case Output:



```
Enter number of elements: 4
Enter 4 elements:
75
45
12
10

Input Case: WORST CASE

Outer loop iterations = 3

Inner loop iterations = 6

Sorted Array:
10 12 45 75
```

Average Case Output:



```
Enter number of elements: 4
Enter 4 elements:
12
41
10
97

Input Case: AVERAGE CASE

Outer loop iterations = 3

Inner loop iterations = 6

Sorted Array:
10 12 41 97
```

Time Complexity:

- Best Case: O(n^2)
- Average Case: O(n^2)
- Worst Case: O(n^2)

Iteration Count:

Case	Outer	Inner
Best	4	10
Average	4	10
Worst	4	10

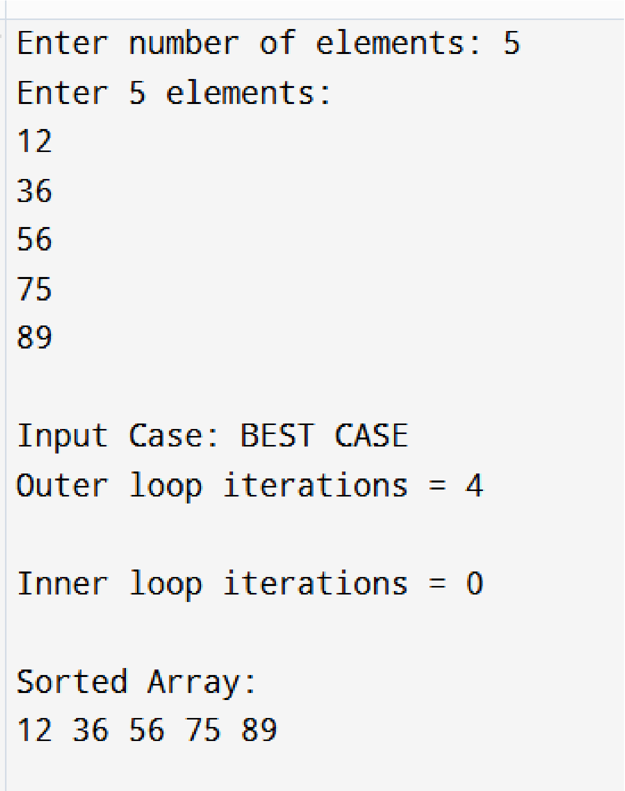
Insertion Sort:

```
#include <stdio.h>
int isSortedAscending(int a[], int n) {
        int i;
  for ( i = 0; i < n - 1; i++) {
    if (a[i] > a[i + 1])
      return 0;
  }
  return 1;
}

int isSortedDescending(int a[], int n) {
        int i;
  for ( i = 0; i < n - 1; i++) {
    if (a[i] < a[i + 1])
      return 0;
  }
  return 1;
}
```

```
void insertionSort(int a[], int n) {
    int i, j, key;
    int outer = 0, inner = 0;

    for (i = 1; i < n; i++) {
        outer++;
        key = a[i];
        j = i - 1;

        while (j >= 0 && a[j] > key) {
            inner++;
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }

    printf("Outer loop iterations = %d\n", outer);
    printf("Inner loop iterations = %d\n", inner);
}

int main() {
    int n,i;
    int a[50];

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for ( i = 0; i < n; i++) {
        scanf("%d", &a[i]);
```

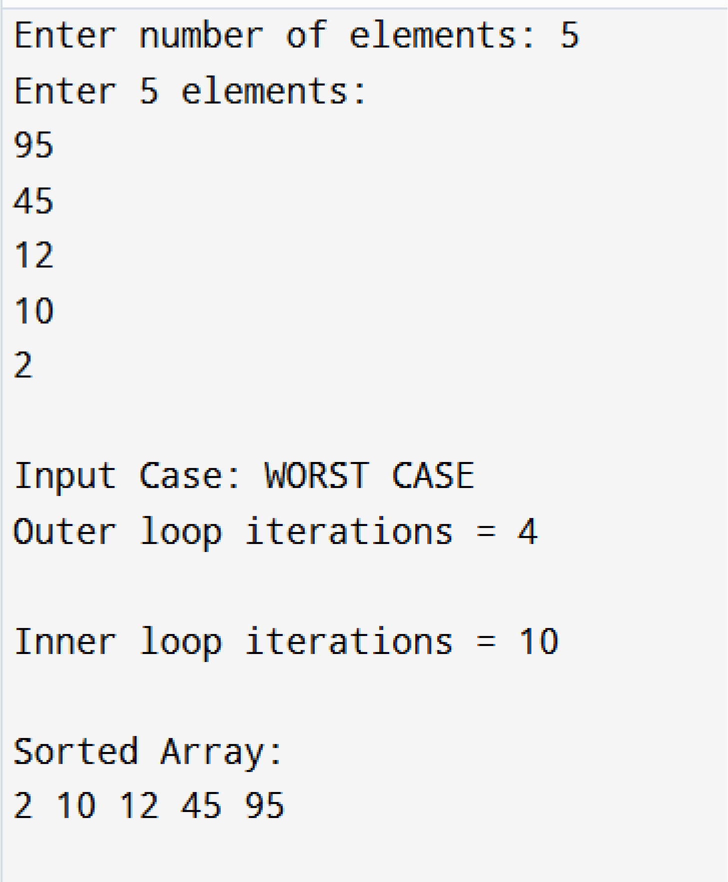
```
  }

  if (isSortedAscending(a, n))
    printf("\nInput Case: BEST CASE");
  else if (isSortedDescending(a, n))
    printf("\nInput Case: WORST CASE");
  else
    printf("\nInput Case: AVERAGE CASE");

  insertionSort(a, n);

  printf("\nSorted Array:\n");
  for (i = 0; i < n; i++)
    printf("%d ", a[i]);

  return 0;
}
```

Best Case Output:



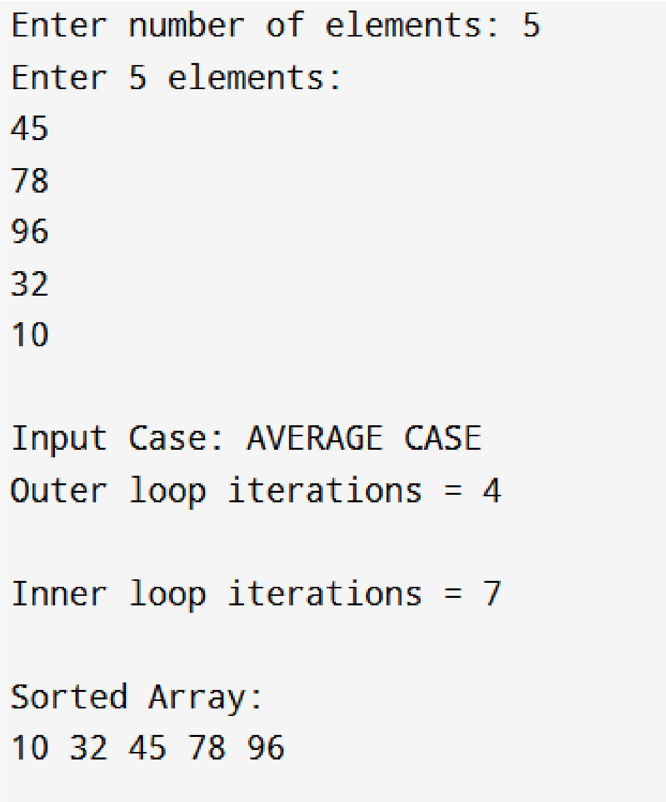
```
Enter number of elements: 5
Enter 5 elements:
12
36
56
75
89

Input Case: BEST CASE
Outer loop iterations = 4

Inner loop iterations = 0

Sorted Array:
12 36 56 75 89
```

Worst Case Output:



```
Enter number of elements: 5
Enter 5 elements:
95
45
12
10
2

Input Case: WORST CASE
Outer loop iterations = 4

Inner loop iterations = 10

Sorted Array:
2 10 12 45 95
```

Average Case Output:



```
Enter number of elements: 5
Enter 5 elements:
45
78
96
32
10

Input Case: AVERAGE CASE
Outer loop iterations = 4

Inner loop iterations = 7

Sorted Array:
10 32 45 78 96
```

Time Complexity:

- Best Case: O(n)
- Average Case: O(n^2)
- Worst Case: O(n^2)

Iteration Count:

Case	Outer	Inner
Best	4	0
Average	4	5
Worst	4	10

Summary Table:

Algorithm	Best Case	Average Case	Worst Case
Bubble Sort	O(n^2)	O(n^2)	O(n^2)
Selection Sort	O(n^2)	O(n^2)	O(n^2)
Insertion Sort	O(n)	O(n^2)	O(n^2)

Conclusion:

- From the analysis, it can be concluded that both Bubble Sort and Selection Sort exhibit a time complexity of $O(n^2)$in the best, average, and worst cases, as their execution time is not influenced by the initial order of the input data.
- In contrast, Insertion Sort performs more efficiently when the input data is already sorted, achieving a best-case time complexity of $O(n)$, although its average and worst-case complexities remain $O(n^2)$.
- Hence, Insertion Sort is better suited for small datasets or data that is nearly sorted when compared to Bubble Sort and Selection Sort.