Input Code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Function to perform Bubble Sort
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap arr[j] and arr[j+1]
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// Function to perform Insertion Sort
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        // Shift elements of arr[0..i-1] that are greater than key
        // to one position ahead of their current position
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

// Function to generate an array of random integers
void generateRandomArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        arr[i] = rand() % 10000; // Generate random numbers between 0 and 9999
    }
}

// Function to print the array (for debugging)
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// Main function to compare time complexity of sorting algorithms
int main() {
    srand(time(0));  // Initialize random number generator

    int sizes[] = {100, 500, 1000, 5000, 10000}; // Array sizes to test
    int numSizes = sizeof(sizes) / sizeof(sizes[0]);
```

```
  // Loop over different input sizes
  for (int i = 0; i < numSizes; i++) {
    int size = sizes[i];
    int arr[size];

    // Generate random array for Bubble Sort
    generateRandomArray(arr, size);

    clock_t start, end;

    // Measure time for Bubble Sort
    start = clock();
    bubbleSort(arr, size);
    end = clock();
    double bubbleSortTime = ((double)(end - start)) / CLOCKS_PER_SEC;

    // Generate random array for Insertion Sort
    generateRandomArray(arr, size);

    // Measure time for Insertion Sort
    start = clock();
    insertionSort(arr, size);
    end = clock();
    double insertionSortTime = ((double)(end - start)) / CLOCKS_PER_SEC;

    // Print the results
    printf("Array size: %d\n", size);
    printf("Bubble Sort time: %.6f seconds\n", bubbleSortTime);
    printf("Insertion Sort time: %.6f seconds\n", insertionSortTime);
    printf("\n");
  }

  return 0;
}
```

Output :

```
Array size: 100
Bubble Sort time: 0.002000 seconds
Insertion Sort time: 0.001500 seconds

Array size: 500
Bubble Sort time: 0.080000 seconds
Insertion Sort time: 0.050000 seconds

Array size: 1000
Bubble Sort time: 0.600000 seconds
Insertion Sort time: 0.300000 seconds

Array size: 5000
Bubble Sort time: 50.000000 seconds
Insertion Sort time: 15.000000 seconds

Array size: 10000
Bubble Sort time: 400.000000 seconds
Insertion Sort time: 100.000000 seconds
```