PRACTICAL 2

AIM: Write a program to sort given elements of an array in ascending order using Bubble sort. Analyze the time complexity for best, average and worst case.

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

int main() {
   int size;
   printf("Enter the Size of array: ");
   scanf("%d", &size);
   int arr[size];
   for (int i = 0; i < size; i++) {
      arr[i] = random() % size;
   }

   clock_t start, end;
   double time_used;

   start = clock();
   for (int i = 0; i < size - 1; i++) {
      for (int j = 0; j < size - i - 1; j++) {
         if (arr[j] > arr[j + 1]) {
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
         }
      }
   }
   end = clock();

   time_used = ((double)(end - start) ) / CLOCKS_PER_SEC;

   printf("Sorted array: \n");
   /*for (int i = 0; i < size; i++) {
      printf("%d ", arr[i]);
   }*/

   printf("\nTime taken: %f seconds\n", time_used);

   clock_t start2, end2;
   double time_used2;
```

```
    start2 = clock() ;
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    end2 = clock() ;
    time_used2 = ((double)(end2 - start2) ) / CLOCKS_PER_SEC;
    printf("\nTime taken by sorted data : %f seconds\n", time_used2);


    clock_t start3, end3;
    double time_used3;
    start3 = clock() ;
    for(int i = size - 2 ; i>=0 ; i--) {
        for(int j= size - i - 2 ; j>=i ; j--){
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    end3= clock() ;
    time_used3 = ((double)(end3 - start3) ) / CLOCKS_PER_SEC;
    printf("\nTime taken by reverse data : %f seconds\n", time_used3);

    printf("\n");
    return 0;
}
```

OUTPUT:

```
harsh_kadecha@Harshs-MacBook-Air DAA % ./Bubble
Enter the Size of array: 100000
Sorted array:

Time taken: 30.430165 seconds

Time taken by sorted data : 10.229077 seconds

Time taken by reverse data : 5.119238 seconds
```

Output for different size of input

	30000	50000	80000	100000	150000
Random Array	1.610712	4.561177	11.826804	30.430165	42.302656
Sorted Array	0.563729	1.562410	4.0039310	10.22790	14.174979
Reverse Sorted Array	0.281864	0.782655	2.000165	5.119238	7.105546

Where elapsed time is in seconds.