Experiment 3

WAP to Perform the Comparative analysis of iterative Sorting algorithm (Bubble sort, Insertion sort, Selection sort)

<u>Bubble sort:</u>

Program:-

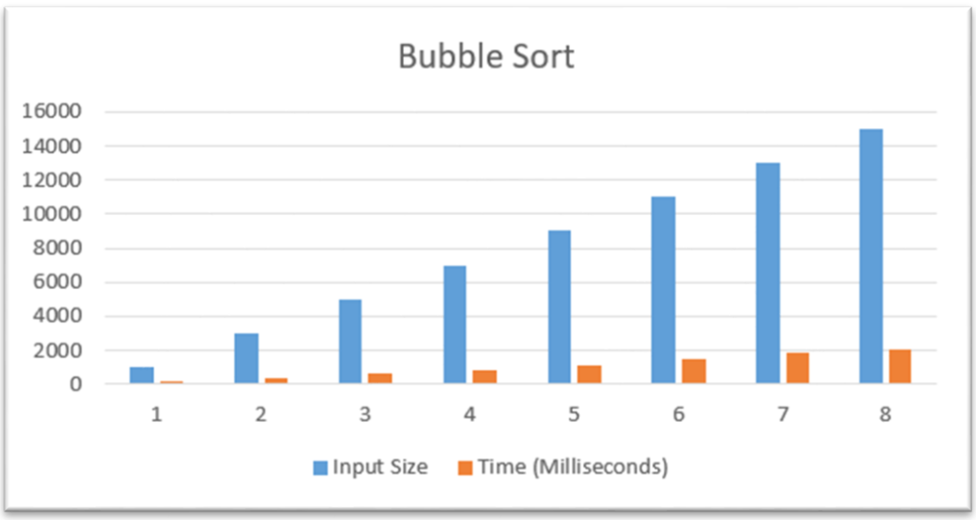
```
#include <stdio.h>
#include <time.h>
#include <conio.h>
int main()
{
    int i,j,n,a[500000], randNum, temp;
    double time;
    clock_t start,end;
    printf("Enter the total input size:");
    scanf("%d",&n);
    start=clock();
    for (i=0;i<n;i++)
    {
        randNum=(rand()%10000);
        a[i]=randNum;
        printf("%d\t", a[i]);
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1;j++)
        {
            if(a[j]>a[j+1])
            {
                a[j]=a[j+1];
                temp=a[j];
                a[j+1]=temp;
            }
        }
    }
    printf("\nThe sorted list:");
    printf("\nThe numbers:");
    for (i=0;i<n;i++;
    {
        printf("%d\t", a[i]);
     }
    end-clock();
    time=((double) (end-start)+1000)/CLOCKS_PER_SEC;
    printf("\n Time %lf millseconds", time);
    return 0;
}
```

Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.81.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Bubble Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	123
3000	346
5000	660
7000	814
9000	1110
11000	1439
13000	1847
15000	2045

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to find the key in it.

Insertion sort

Program:-

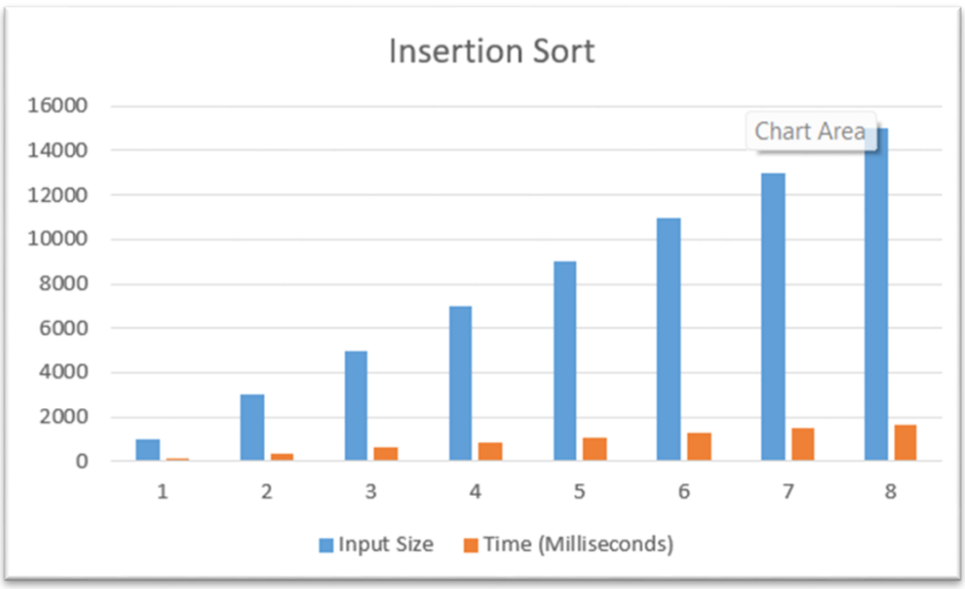
```
#include <stdio.h>
#include <conio.h>
#include <time.h>
int main()
{
    int n, i, j, a[5000], randnum, temp;
    double time;
    clock_t start, end ;
    printf("enter the number of elements");
    scanf("%d", &n);
    start = clock();
    for (i = 0; i < n; i++)
    {
        randnum = (rand() % 10000);
        a[i] = randnum;
        printf("%d\t", a[i]);
    }
    for (i = 1; i < n - 1; i++)
    {
        temp = a[i];
        j = i - 1;
        while (temp <= a[j] && (j >= 0))
        {
            a[j + 1] = a[j];
            j = j - 1;
        }
        a[j + 1] = temp;
    }
    printf("\nsorted list:");
    for (i = 0; i < n; i++)
    {
    printf("%d\t", a[i]);
    }
  end = clock();
  time = ((double)(end - start) * 1000) / CLOCKS_PER_SEC;
  printf("\ntime=%lf milli seconds", time);
  return 0;
}
```

Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.81.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Insertion Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	124
3000	317
5000	661
7000	854
9000	1090
11000	1254
13000	1530
15000	1618

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to find the key in it.

<u>Selection sort</u>

Program:-

```
#include <stdio.h>
#include <time.h>
#include <conio.h>
int smallest(int a[], int k, int n, int pos) {
  int small, j;
  small = a[k];
  for (j = k + 1; j < n - 1; j++) {
     if (small > a[j]) {
        small = a[j];
        pos = j;
     }
  }
  return pos;
}
int main() {
  int i, j, n, a[500000], randNum, temp, pos;
  double time;
  clock_t start, end;
  printf("Enter the total input size:");
  scanf("%d", &n);
  start = clock();
  for (i = 0; i < n; i++) {
     randNum = (rand() % 10000);
     a[i] = randNum;
     printf("%d\t", a[i]);
  }
  for (i = 1; i < n - 1; i++) {
     pos = smallest(a, i, n, pos);
```

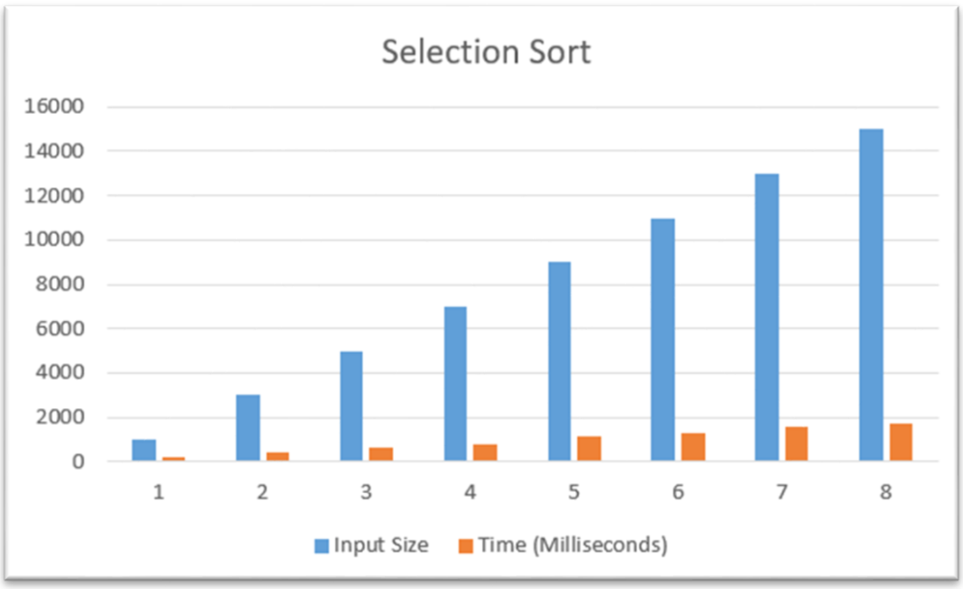
```
        temp = a[i];
        a[i] = a[pos];
        a[pos] = temp;
    }
    printf("\nThe sorted list:\nThe numbers:");
    for (i = 0; i < n; i++) {
        printf("%d\t", a[i]);
    }
    end = clock();
    time = ((double)(end - start) * 1008) / CLOCKS_PER_SEC;
    printf("\nTime=%lf milliseconds", time);
    return 0;
}
```

Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.81.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Selection Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	169
3000	379
5000	611
7000	798
9000	1145
11000	1304
13000	1537
15000	1717

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to find the key from the array.

Conclusion

In this experiment, it has been found that the size of input "n" has linear relationship with the time taken by the system to sort the array. On comparing these three sorting algorithm, it has been found that the Bubble Sort is the slowest and Insertion Sort is the fastest to sort an array and Selection sort falls in between. So, Insertion sort is better than other sorting algorithm to sort an array.