Experiment 7

WAP to Perform the Comparative analysis of Naïve algorithm and Randomized selection algorithm to solve selection problem.

Randomized selection

Program:-

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int randomized_partition(int a[], int low, int high) {
    int pivot_index = low + rand() % (high - low + 1);
    int temp = a[pivot_index];
    a[pivot_index] = a[high];
    a[high] = temp;
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (a[j] <= a[high]) {
            i++;
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
    temp = a[i + 1];
    a[i + 1] = a[high];
    a[high] = temp;

    return i + 1;
}
void randomized_selection_sort(int a[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int k = i; // Find the i-th smallest element
```

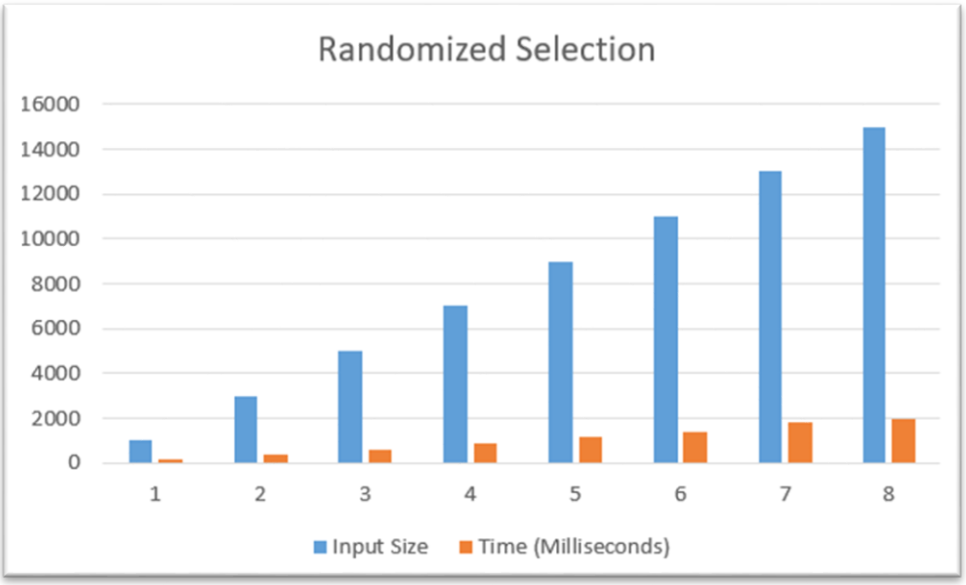
```
        int pos = randomized_partition(a, k, n - 1);
        if (pos != k) {
            int temp = a[k];
            a[k] = a[pos];
            a[pos] = temp;
        }
    }
}
int main() {
int i, j, n, a[500000], randNum, temp, pos;
    double time;
    clock_t start, end;
    printf("Enter the 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]);
    }
    randomized_selection_sort(a, n);
    printf("\nThe sorted list:\nThe numbers:");
    for (i = 0; i < n; i++) {
        printf("%d\t", a[i]);
    }
    end = clock();
    time = ((double)(end - start) * 1000) / 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 1.85.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Randomized selection algorithm 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	345
5000	605
7000	878
9000	1160
11000	1401
13000	1842
15000	1986

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 sort an array.

Naïve algorithm

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
   int i, j, n, a[500000], randNum, temp, min_index;
   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 - 1; i++) {
      min_index = i;
      for (j = i + 1; j < n; j++) {
         if (a[j] < a[min_index]) {
            min_index = j;
         }
      }
      temp = a[i];
      a[i] = a[min_index];
      a[min_index] = temp;
   }

   printf("\nThe sorted list:\nThe numbers:");
```

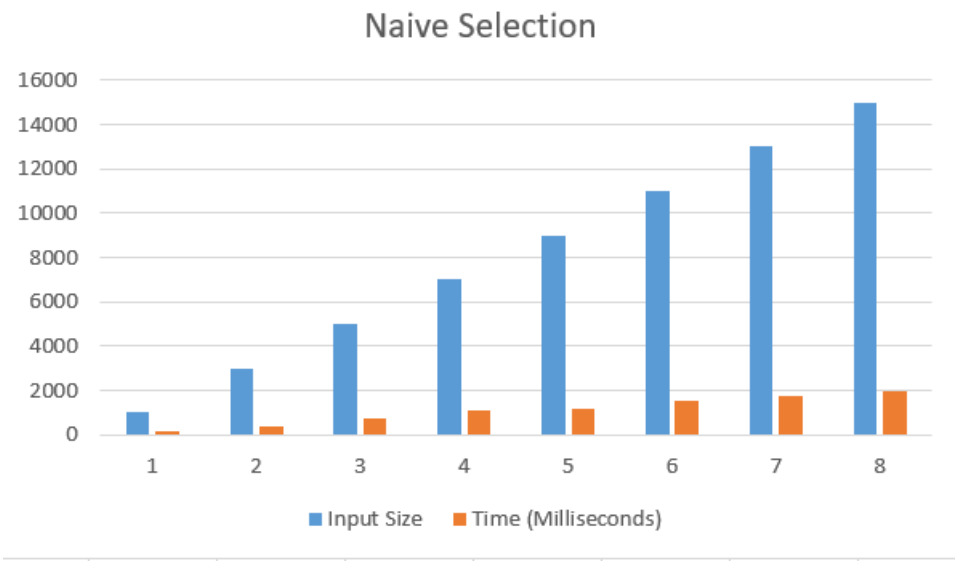
```
    for (i = 0; i < n; i++) {
        printf("%d\t", a[i]);
    }
    end = clock();
    time = ((double)(end - start) * 1000) / CLOCKS_PER_SEC;
    printf("\nTime=%lf milliseconds", time);
    return 0;
}
```

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 1.85.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Naïve algorithm 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	134
3000	410
5000	718
7000	1077
9000	1155
11000	1523
13000	1729
15000	1964

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 sort an 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 .So, Comparatively Naïve selection is better than Randomized selection algorithm to sort a large array whereas Randomized Selection is good for small size of array.