Ankit Raj 1906534

Q1. write a program to perform linear search operation in an array of n integers. Determine the time required to search an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n. The n integers can be generated randomly.

```
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
int LinearSearch(int arr[], int n, int p){
for (int i = 0; i < n; i++){
  if (arr[i] == p)
        return i;
   }
     return -1;
}

int main()
{
clock_t start, end;
double total_cputime;
int n;
printf("Enter size of array: ");
scanf("%d",&n);
```

```
int a[n];
start = clock();
for(int i=0;i a[i]=(rand()%1000);
    printf("Elements are successfully generated randomly\n");
int x=rand()%200;
  printf("Number to be searched: %d\n",x);
int result=LinearSearch(a,n,x);
(result == -1)   ?printf("Element is not present in array")   :
 printf("Element is present at index %d", result);
end = clock();

printf("\n\ncpu Time calculation:");
printf("\nStarting time:: %ld",start);
printf("\nEnd time:: %ld",end);
total_cputime = ((double)(end - start));
 printf("\nTotal CPU time:: %f",total_cputime);
 total_cputime = ((double)(end - start))/CLOCKS_PER_SEC;
 printf("\nTotal CPU time:: %f",total_cputime);
 return 0;
}
```

Q2. write a program to sort a given set of elements using the insertion sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include <stdio.h>
#include<stdlib.h>
#include<time.h>
void insertionsort(int arr[], int n) {
int i, j;
for(i=1;i int temp = arr[i];
j = i-1;
while(arr[j] > temp && j>=0) {
    arr[j+1] = arr[j];
    j--;
     }
    arr[j+1] = temp;
     }
}

int main() {
clock_t start, end;
double total_cputime;
int n, i, j;
printf("Enter size of array:: ");
scanf("%d",&n);
int arr[n];
start = clock();
for(i=0;i arr[i]=(rand());
printf("%d\t",arr[i]);
}
printf("\nRandom numbers generated successfully...\n\n");

insertionSort(arr, n);
printf("Array after insertion sort:: \n");
for(i=0;i printf("%d\t",arr[i]);
      }
end = clock();
 printf("\n\nCPU time calculation:: ");
 printf("\nStarting time: %ld ms",start);
 printf("\nEnding time: %ld ms",end);
 total_cputime = ((double)(end - start));
 printf("\nTotal CPU time (in ms): %f ms",total_cputime);
 total_cputime = ((double)(end - start))/CLOCKS_PER_SEC;
  printf("\nTotal CPU time (in s): %f s",total_cputime);
 return 0;
}
```