Practical-2

(1) Generate large number of elements randomly and sort all the elements in Ascending order using Selection sort. Analyse the time complexity for best, average and worst case.

Code:

```
#include<stdio.h>
#include<time.h>
#include<sys/time.h>
int a[20000],i,n;
void selectionsort()
{
   int j,index,temp,max=0;
   for(i=n-1;i>=0;i--)
   {
      max=0;
      for(j=0;j<=i;j++)
      {
         if(a[j]>max)
         {
            max=a[j];
            index=j;
         }
      }
      temp=a[i];
      a[i]=a[index];
      a[index]=temp;
   }
}
void checkfor()
{
   int t2,t1;
   struct timeval tv;
   struct timezone tz;

   //Best Case
```

```
    for(i=0;i<n;i++)
    {
       a[i]=i;
    }
    gettimeofday(&tv,&tz);
    t1=((tv.tv_sec*1000000)+(tv.tv_usec));
    selectionsort();
    gettimeofday(&tv,&tz);
    t2=((tv.tv_sec*1000000)+(tv.tv_usec));
    printf("\n %d \t\t %d ",n,(t2-t1));

    //Average case

    for(i=0;i<n;i++)
    {
       a[i]=rand()%n;
    }
    gettimeofday(&tv,&tz);
    t1=((tv.tv_sec*1000000)+(tv.tv_usec));
    selectionsort();
    gettimeofday(&tv,&tz);
    t2=((tv.tv_sec*1000000)+(tv.tv_usec));
    printf("\t\t%d",(t2-t1));

    //worst case

    for(i=0;i<n;i++)
    {
       a[i]=n-i;
    }
    gettimeofday(&tv,&tz);
    t1=((tv.tv_sec*1000000)+(tv.tv_usec));
    selectionsort();
    gettimeofday(&tv,&tz);
    t2=((tv.tv_sec*1000000)+(tv.tv_usec));
    printf("\t\t%d",(t2-t1));
}
void main()
```

```
{
 printf("\n Value     \tBest case\tAverage case\tWorst case\n");
 n=5000;
 checkfor();
 n=10000;
 checkfor();
 n=15000;
 checkfor();
 n=20000;
 checkfor();
}
```

Output-Table:

Value	Best Case	Average Case	Worst Case
5000	174976	112750	187391
10000	694627	512027	605346
15000	1499456	1201232	1328287
20000	2777729	2199399	2308905

Graph:

(2) Generate large number of elements randomly and sort all the elements in ascending order using Insertion sort. Analyse the time complexity for best, average and worst case.

Code:

```
#include<stdio.h>
#include<time.h>
#include<sys/time.h>
int a[20000],i,n;
void insertionsort()
{
   int j,temp;
   for(i=1;i<n;i++)
   {
      temp=a[i];
      j=i-1;
      while(j>=0 && a[j]>temp)
      {
         a[j+1]=a[j];
         j=j-1;
      }
      a[j+1]=temp;
   }
 /* printf("\n");
   for(i=0;i<n;i++)
      printf("%d\t",a[i]); */
}
void checkfor()
{
   int t2,t1;
   struct timeval tv;
   struct timezone tz;

   //Best Case

   for(i=0;i<n;i++)
```

```
    {
       a[i]=i;
    }
    gettimeofday(&tv,&tz);
    t1=((tv.tv_sec*1000000)+(tv.tv_usec));
    insertionsort();
    gettimeofday(&tv,&tz);
    t2=((tv.tv_sec*1000000)+(tv.tv_usec));
    printf("\n %d \t\t %d ",n,(t2-t1));

    //Average case

    for(i=0;i<n;i++)
    {
       a[i]=rand()%n;
    }
    gettimeofday(&tv,&tz);
    t1=((tv.tv_sec*1000000)+(tv.tv_usec));
    insertionsort();
    gettimeofday(&tv,&tz);
    t2=((tv.tv_sec*1000000)+(tv.tv_usec));
    printf("\t\t%d",(t2-t1));

    //worst case

    for(i=0;i<n;i++)
    {
       a[i]=n-i;
    }
    gettimeofday(&tv,&tz);
    t1=((tv.tv_sec*1000000)+(tv.tv_usec));
    insertionsort();
    gettimeofday(&tv,&tz);
    t2=((tv.tv_sec*1000000)+(tv.tv_usec));
    printf("\t\t%d",(t2-t1));
}
void main()
{
```

```
 printf("\n Value     \tBest case\tAverage case\tWorst case\n");
 n=5000;
 checkfor();
 n=10000;
 checkfor();
 n=15000;
 checkfor();
 n=20000;
 checkfor();
}
```

Output-Table:

Value	Best Case	Average Case	Worst Case
5000	30	80190	190203
10000	48	308980	691709
15000	77	716064	1489796
20000	77	1398269	2813310

Graph: