WEEK-1

AIM: Implement Selection sort and find how many steps are required to sort 10 elements.

CODE:

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

int SelectionSort(int arr[],int n){
    int i,j,c=0,t,m;
    for (i = 0; i< n - 1; i++) {c++;
      m = i;c++;
      for (j = i + 1; j < n; j++) {c++;
        if (arr[j] <arr[m]) {
          m = j;c++;
}c++;
      }
      if (m != i) {
          t = arr[m];c++;
arr[m] = arr[i];c++;
arr[i] = t;c++;
}c++;
    }
  return c+2;
}
int main(){
    int *arr,n,i,c,count;
printf("Enter the size of the array: ");
scanf("%d",&n);
printf("Enter 1 for best case scenario\t2 for average case
scenario\t3 for worst case scenario: ");
scanf("%d",&c);
arr=(int*)malloc(n*sizeof(int));
    switch (c)
    {
    case 1:
        for(i=0;i<n;i++){
```

```
arr[i]=i+1;
        }
break;
    case 2:
srand(time(0));
        for ( i = 0; i< n; i++)
        {
arr[i]=rand()%n+1;
        }
break;
    case 3:
        for ( i = 0; i< n; i++)
        {
arr[i]=n-i;
        }
break;
    default:
printf("Incorrect Input");
break;
    }
    /*for ( i = 0; i< n; i++)
    {
printf("%d  ",arr[i]);
    }
printf("\n");
start_t=clock();*/
    count=SelectionSort(arr,n);
printf("Step Count : %d steps\n",count);
}
```

Sample Output:

```
Enter the size of the array: 10

Enter 1 for best case scenario   2 for average case scenario     3
for worst case scenario: 1

Step Count: 315 steps
```

WEEK-2

AIM: Implement and Analysis factorial of a number program using iterative and recursive methods

CODE:

1. Using Iterative method:

```
#include<stdio.h>
int c=0;
int fact(int n){


    int fact=1;
    while (n>1)
    {
        fact=fact*n;c++;
        n--;c++;
c++;
    }
c++;
c++;
    return fact;
}
int main() {
    int n1, n2;
printf("Enter a positive integer: ");
scanf("%d", &n1);
printf("Factorial of %d is %d.", n1, fact(n1));
printf("\nStep Count: %d",c);
    return 0;
}
```

Sample Output:

```
Enter a positive integer: 10
Factorial of 10 is 3628800.
Step Count: 29

Enter a positive integer: 5
Factorial of 5 is 120.
Step Count: 14
```

2. Using Recursive Method:

```
#include<stdio.h>
int c=0;
int fact(int n){
    if (n>1)
    {
        c=c+2;
        return n*fact(n-1);
    }
    else
    {
        c=c+3;
        return 1;
    }
}
int main() {
    int n1, n2;
printf("Enter a positive integer: ");
scanf("%d", &n1);
printf("Factorial of %d is %d.", n1, fact(n1));
printf("\nStep Count: %d",c);
    return 0;
}
```

Sample Output:

```
Enter a positive integer: 10
Factorial of 10 is 3628800.
Step Count: 21

Enter a positive integer: 5
Factorial of 5 is 120.
Step Count: 11
```

WEEK-3

AIM: Implement Insertion Sort and analyse the time complexity

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int InsertionSort(int arr[],int n){
    int i,j,t,c=1;
    for (i = 1; i< n; i++,c++) {
      t = arr[i];c++;
for( j = i;j> 0 && t <arr[j - 1];j--,c++){
arr[j] = arr[j - 1];c++;
      }
arr[j] = t;c++;
    }
    return c+2;

}
int main(){
    int *arr,n,i,c,count;
clock_tstart_t,end_t,total_t;
printf("Enter the size of the array: ");
scanf("%d",&n);
printf("Enter 1 for best case scenario\t2 for average case 
scenario\t3 for worst case scenario: ");
scanf("%d",&c);
arr=(int*)malloc(n*sizeof(int));
    switch (c)
    {
    case 1:
        for(i=0;i<n;i++){
arr[i]=i+1;
        }
break;
    case 2:
srand(time(0));
        for ( i = 0; i< n; i++)
        {
arr[i]=rand()%n+1;
```

```
        }
break;
    case 3:
        for ( i = 0; i< n; i++)
        {
arr[i]=n-i;
        }
break;
    default:
printf("Incorrect Input");
break;
    }
    /*for ( i = 0; i< n; i++)
    {
printf("%d  ",arr[i]);
    }
printf("\n");
start_t=clock();*/
    count=InsertionSort(arr,n);
    /*end_t=clock();
total_t=end_t - start_t;
    for ( i = 0; i< n; i++)
    {
printf("%d  ",arr[i]);
    }

printf("\nProcessor cycles taken: %lld cycles\n",(long
long)total_t);*/
printf("Step Count: %d steps\n",count);
}
```

Sample Output:

```
Enter the size of the array: 12

Enter 1 for best case scenario   2 for average case scenario     3
for worst case scenario: 1

Step Count: 36 steps
```

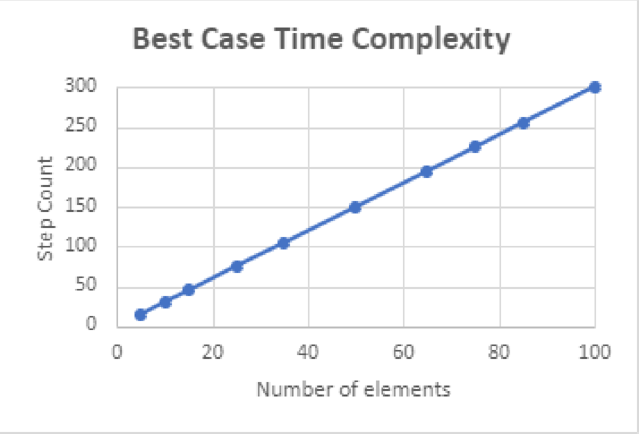
Time Complexity Analysis:

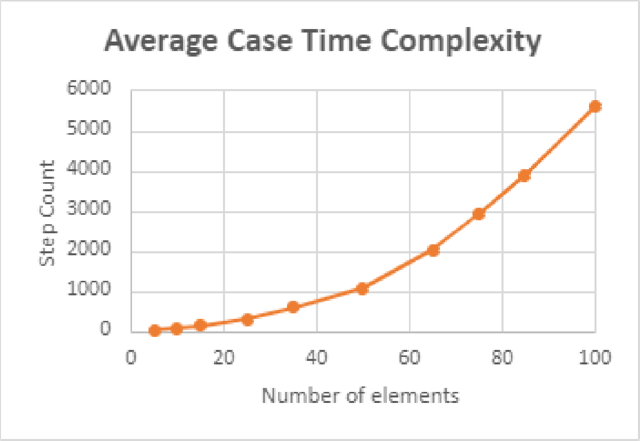
Best Case Time Complexity = **O(n)**

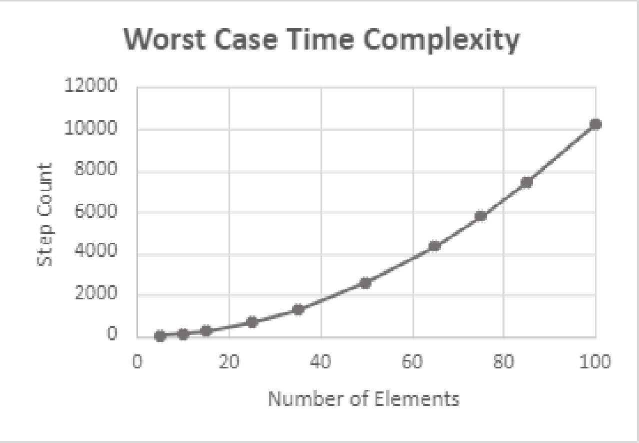
Average Case Time Complexity = **O(n^2)**

Worst Case Time Complexity = **O(n^2)**

N	Step Count		
	Best Case	Average Case	Worst Case
5	34	38	46
10	119	149	159
15	254	309	331
25	674	776	866
35	1294	1442	1651
50	2599	2873	3299
65	4354	4721	5506
75	5774	6217	7291
85	7394	7850	9326
100	10199	10774	12849







WEEK-4

AIM: Given two strings, find the minimum number of edits required to convert one string to another

CODE:

```
#include<stdio.h>
#include<string.h>
int min(int x,int y,int z){return x<y && x<z?x:y<z?y:z;}
int Min_Operations(char A[],char B[],int m,int n){
   if(m==0) return n;
   if(n==0) return m;
   if (A[m - 1] == B[n - 1])
      return Min_Operations(A, B, m - 1, n - 1);
   return 1 + min(Min_Operations(A, B, m, n - 1),  // Insert
            Min_Operations(A, B, m - 1, n),     // Remove
            Min_Operations(A, B, m - 1,n - 1)); // Replace
}
int main(){
   char A[40],B[40];
   printf("Enter String 1 and String 2:");
   scanf("%s%s",A,B);
   printf("\nString 1 is %s and String 2 is %s\nThe minimum no of steps required to edit a
string is %d\n",A,B,Min_Operations(A,B,strlen(A),strlen(B)));
}
```

OUTPUT:

```
Enter string 1 and string 2:hari
gopi
```

string 1 is hari and string 2 is gopi

The minimum no of steps requires to edit the string is 3

WEEK-5

AIM: Write a program to find the Greatest Common Divisor of two numbers using recursion and find how many steps are required to execute it

CODE:

```
int C=0;
#include <stdio.h>
int gcd(int n1, int n2) {
    if (n2 != 0){
        C=C+2;
        return gcd(n2, n1 % n2);}
else{
        C=C+3;
        return n1;}
}
int main() {
    int n1, n2;
printf("Enter two positive integers: ");
scanf("%d %d", &n1, &n2);
printf("G.C.D of %d and %d is %d.", n1, n2, gcd(n1, n2));
printf("Step Count: %d",C);
    return 0;
}
```

Sample Output:

```
Enter two positive integers: 19  17
G.C.D of 19 and 17 is 1.
Step Count: 9
```

```
Enter two positive integers: 1547 1554
G.C.D of 1547 and 1554 is 7.
Step Count: 9
```

WEEK-6

AIM: Sort a given set of elements using the quick 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

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
int partition(int array[], int low, int high) {
  int pivot = array[high];
  int i = (low - 1);
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {
i++;
      swap(&array[i], &array[j]);
    }
  }
swap(&array[i + 1], &array[high]);
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {
    int pi = partition(array, low, high);
quickSort(array, low, pi - 1);
quickSort(array, pi + 1, high);
  }
}
int main(){
    int *arr,n,i,c,count;
clock_tstart_t,end_t;
```

```
    //double total_t;
    double total_t;
printf("Enter the size of the array: ");
scanf("%d",&n);
printf("Enter 1 for best case scenario\t2 for average case
scenario\t3 for worst case scenario: ");
scanf("%d",&c);
arr=(int*)malloc(n*sizeof(int));
    switch (c)
    {
    case 1:
        for(i=0;i<n;i++){
arr[i]=i+1;

        }
break;
    case 2:
srand(time(0));
        for ( i = 0; i< n; i++)
        {
arr[i]=rand()%n+1;
        }
break;
    case 3:
        for ( i = 0; i< n; i++)
        {
arr[i]=n-i;
        }
break;
    default:
printf("Incorrect Input");
break;
    }
    /*for ( i = 0; i< n; i++)
    {
printf("%d  ",arr[i]);
    }
printf("\n");*/
start_t=clock();
quickSort(arr,0,n-1);
end_t=clock();
total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;
    /*for ( i = 0; i< n; i++)
    {

printf("%d  ",arr[i]);

    }*/
    //printf("\nProcessor cycles taken : %Lf cycles\n",total_t);
printf("\nThe time taken is %lf seconds",total_t);
    //printf("Step Count : %d steps\n",c);

}
```

Sample Output:

```
Enter the size of the array: 1000
Enter 1 for best case scenario  2 for average case scenario     3
for worst case scenario: 1

The time taken is 0.002982 seconds
```

WEEK-7

AIM: Write a program to check whether a given graph is connected or not using the DFS method

CODE:

```
#include<stdio.h>
int visit[20],n,adj[20][20],s,count=0;

void dfs(int v)
{
  int w;
  visit[v]=1;
  count++;
  for(w=1;w<=n;w++)

    if((adj[v][w]==1) && (visit[w]==0))
      dfs(w);
}
void main()
{
  int v,w;
  printf("Enter the no.of vertices:");
  scanf("%d",&n);

  printf("Enter the adjacency matrix:\n");
  for(v=1;v<=n;v++)
    for(w=1;w<=n;w++)
      scanf("%d",&adj[v][w]);

  for(v=1;v<=n;v++)
      visit[v]=0;

    dfs(1);

  if(count==n)
   printf("\nThe graph is connected");
  else
   printf("The graph is not connected");
}
```

Sample Output:

```
1.Enter no of vertices : 3
  Enter adjacency matrix : [ 1 1 0
                             1 0 0
                             0 0 0 ]


  1->2
  The Graph is not connected
```