Nandini
DATE .....
PAGE .....

BUBBLE SORT

```
#include<stdio.h>
#include<time.h>
int main()
{
int array[100],n,c,d,swap;
printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enter %d integers\n",n);
for(c=0;c<n;c++)
scanf("%d",&array[c]);
clock_t start end;
double cpu_time_used;
start = clock();
for(c=0;c<n-1;c++)
{
for(d=0;d<n-c-1;d++)
{
if(array[d]>array[d+1])
{
swap=array[d];
array[d]=array[d+1];
array[d+1]=swap;
}
}
}
```

Nandini
DATE .....
PAGE .....

```
printf("Sorted list in ascending order:\n");
for(c=0;c<n;c++)
printf("%d\n",array[c]);
end=clock();
printf("time taken is %f", ((double)(end-start))/
                                clocks_per_sec);
return 0;
}
```

Nandini
DATE .....
PAGE .....

BINARY SEARCH

```
#include<stdio.h>
#include<time.h>
int main()
{
int c,f,l,m,n,search,array[100];
printf("Enter the number of elements\n");
scanf("%d",&n);
printf("Enter %d integers\n",n);
for(c=0;c<n;c++)
scanf("%d",&array[c]);
printf("Enter value to find\n");
scanf("%d",&search);
f=0;
l=n-1;
m=(f+l)/2;
clock_t start,end;
double cpu_time_used;
start=clock();
while(f<l)
{
if(array[m]==search)
{
printf("%d found at location %d.\n",search,m+1);
break;
}
```

Nandini
DATE .....
PAGE .....

MERGE SORT

```
#include<stdio.h>
#include<time.h>
void mergesort(int a[], int i, int j);
void merge(int a[], int i1, int i2, int j1, int j2);
int main()
{
int a[30], n, i;
printf("Enter no. of elements");
scanf("%d", &n);
printf("Enter array elements");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
clock_t start, end;
double cpu_time_used;
start=clock();
mergesort(a, 0, n-1);
end= clock();
printf("\n Sorted array is:");
for(i=0; i<n; i++)
printf("%d", a[i]);
printf("Total time taken is %f", ((double)(end-start))/
                                  CLOCKS_PER_SEC);
return 0;
}
```

```
void mergesort(int a[], int i, int j)
{
 int mid;
 if(i<j)
 mid=(i+j)/2;
 mergesort(a, i, mid);
 mergesort(a, mid+1, j);
 merge(a, i, mid, mid+1, j);
 }
}
void merge(int a[], int i1, int j1, int i2, int j2)
{
 int temp[50];
 int i, j, k;
 i=i1;
 j=i2;
 k=0;
while(i<=j1 && j<=j2)
{
 if(a[i]<a[j])
 temp[k++]=a[i++];
 else
 temp[k++]=a[j++];
}
while(i<=j1)
 temp[k++]=a[i++];
while(j<=j2)
```

```
temp[k++] = a[j++];
for (i = i1, j = 0; i <= j2; i++, j++)
a[i] = temp[j];
}
```

Nandini
DATE .....
PAGE .....

QUICK SORT

```
#include <stdio.h>
#include<time.h>
void quicksort(int[10], int, int);
int main()
{
int x[20], size, i;
printf("Enter size of the array: ");
scanf("%d", &size);
printf("Enter %d elements: ",size);
for(i=0; i<size; i++)
scanf("%d", &x[i]);
clock_t start, end;
double cpu_time_used;
start = clock();
quicksort(x,0,size-1);
printf("Sorted Elements: ");
for(i=0; i<size; i++)
printf("%d", x[i]);
end = clock();
printf("Time taken: %f", ((double)(start-end))/(clocks_per_sec));
return 0;
}
void quicksort(int x[10], int first, int last)
{
int pivot, j, temp, i;
```

Nandini
DATE .....
PAGE .....

```
if ( first < last)
{
pivot= first;
i= first;
j = last;
while (i<j)
{
while (x[i]<=x[pivot] && i< last)
i++;
while (x[j] >x[pivot])
j--;
if (i<j)
{
temp = x[i];
x[i] = x[j];
x[j]= temp;
}}
temp= x[pivot];
x[pivot]= x[j];
x[j]= temp;
quicksort (x, first, j-1);
quicksort (x, j+1, last);
}
}
```

Nandini
DATE .....
PAGE .....

KNAPSACK PROBLEM

```
#include <stdio.h>
void knapsack(int n, float w[], float profit[], float
capacity) {
float x[20], tp=0
int i, weight ;
weight=0;
for(i=0; i<n; i++)
{
 x[i]=0.0;
for(i= weight ; i< capacity ; i++)
{
  if(weight + w[i] <=capacity)
{
  x[i]=1.0;
  weight = weight+ w[i];
 }
else
{
x[i]= (capacity -weight)/w[i];
weight = capacity;
}
}
}

for(i=0; i<n; i++)
 tp= tp+ (x[i]*profit[i]);
```

```
printf("\n The result vector is :- ");
for(i=0; i<n; i++)
printf("%f\t", x[i]);
printf("\n Maximum profit is :- %f", tp);
}
int main()
{
float weight[20], profit[20], capacity;
int num, i, j;
float ratio[20], temp;
printf("\n Enter the wts. & profits of each object :- ");
for(i=0; i<num; i++)
{
scanf("%f %f", &weight[i], &profit[i]);
}
printf("\n Enter the capacity of knapsack :- ");
scanf("%f", &capacity);
for(i=0; i<num; i++){
ratio[i] = profit[i]/weight[i];
}
for(i=0; i<num; i++){
for(j=i+1; j<num; j++){
if(ratio[i]<ratio[j]){
temp = ratio[j];
ratio[j] = ratio[i];
ratio[i] = temp;
```

Nandini
DATE .....
PAGE .....

```
temp = weight[j];
weight[j] = weight[i];
weight[i] = temp;

temp = profit[j];
profit[j] = profit[i];
profit[i] = temp;
}
}
}

knapsack(num, weight, profit, capacity);
return 0;
}
```

Nandini
DATE .....
PAGE .....

Insertion Sort

```
#include<stdio.h>
int main()
{
int i,j,n,temp,a[30];
printf("Enter no. of elements");
scanf("%d",&n);
printf("\n Enter the elements\n");
for(i=0;i<n;i++)
{
 scanf("%d",&a[i]);
}
for(i=1;i<=n-1;i++)
 {
   temp=a[i];
   j=j-1;
   while((temp<a[j])&&(j>=0))
{
   a[j+1]=a[j];
     j=j-1;
}
   a[j+1]=temp;
} printf("Sorted list is\n");
for(i=0;i<n;i++)
{ printf("%d",a[i]);
} return 0;
}
```

18.11.12

KRUSKAL'S CODE

Prashansa Gupta
CCVT-B3
112

```
#include<stdio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min, mincost=0, cost[9][9], parent[9];
int find(int);
int uni(int,int);
void main()
{
printf("\n\t Implementation of Kruskal's Algo \n");
printf("\nEnter the no. of vertices: ");
scanf("%d",&n);
printf("\n Enter the cost adjacency matrix: \n");
for(i=1; i<=n; i++)
{
  for(j=1; j<=n; j++)
  {
    scanf("%d", &cost[i][j]);
    if(cost[i][j]==0)
      cost[i][j]=999;
  }
}
printf("The edges of Minimum cost spanning Tree are \n");
while(ne<n)
{ for(i=1; min=999; i<=n; i++)
  { for(j=1; j<=n; j++)
    { if(cost[i][j]<min)
      { min=cost[i][j];
        a=u=i;
        b=v=j;
      } } }
  u=find(u);
  v=find(v);
  if(uni(u,v))
```

```
    {
      printf("%d edge (%d,%d) = %d\n", ne++, a, b, min);
      mincost += min;
    }
    cost[a][b] = cost[b][a] = 999;
  }
  printf("\n\tMinimum cost = %d\n", mincost);
}
int find(int i)
{
  while (parent[i])
  i = parent[i];
  return i;
}
int uni(int i, int j)
{
  if(i != j)
  {
    parent[j]=i;
    return 1;
  }
  return 0;
}
```