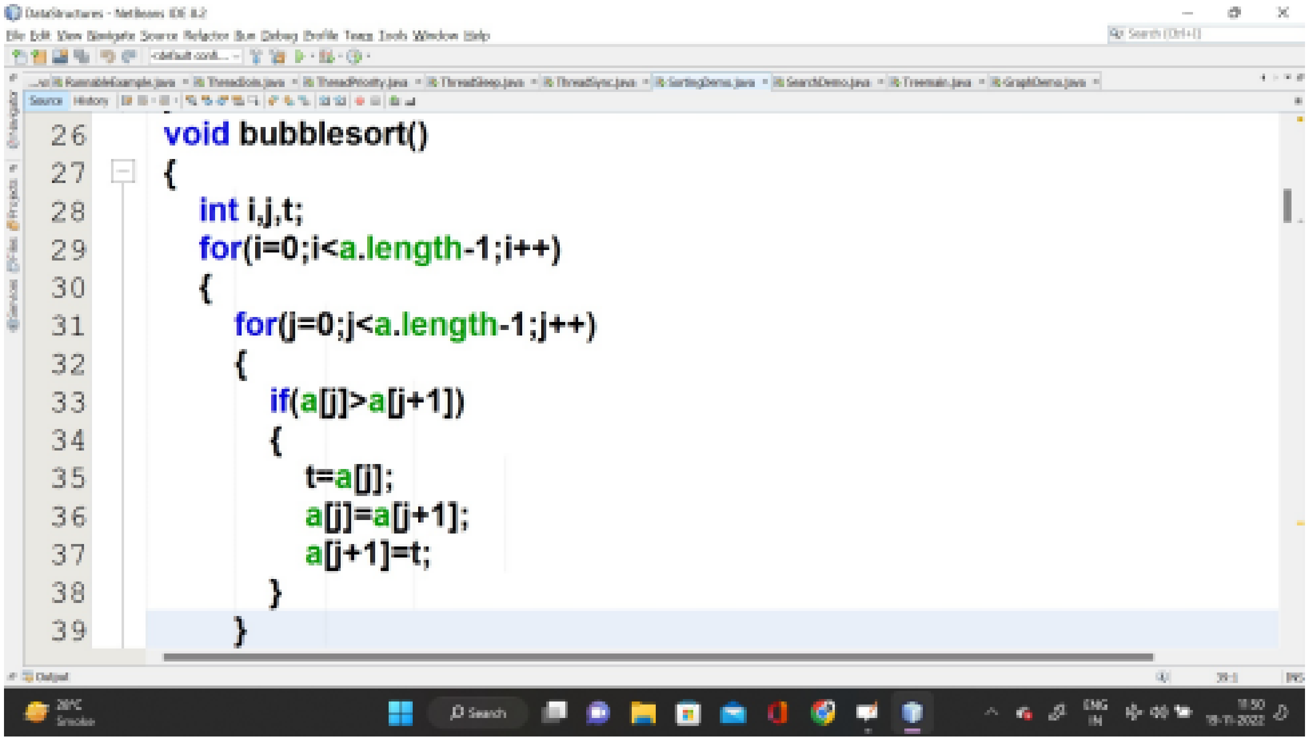
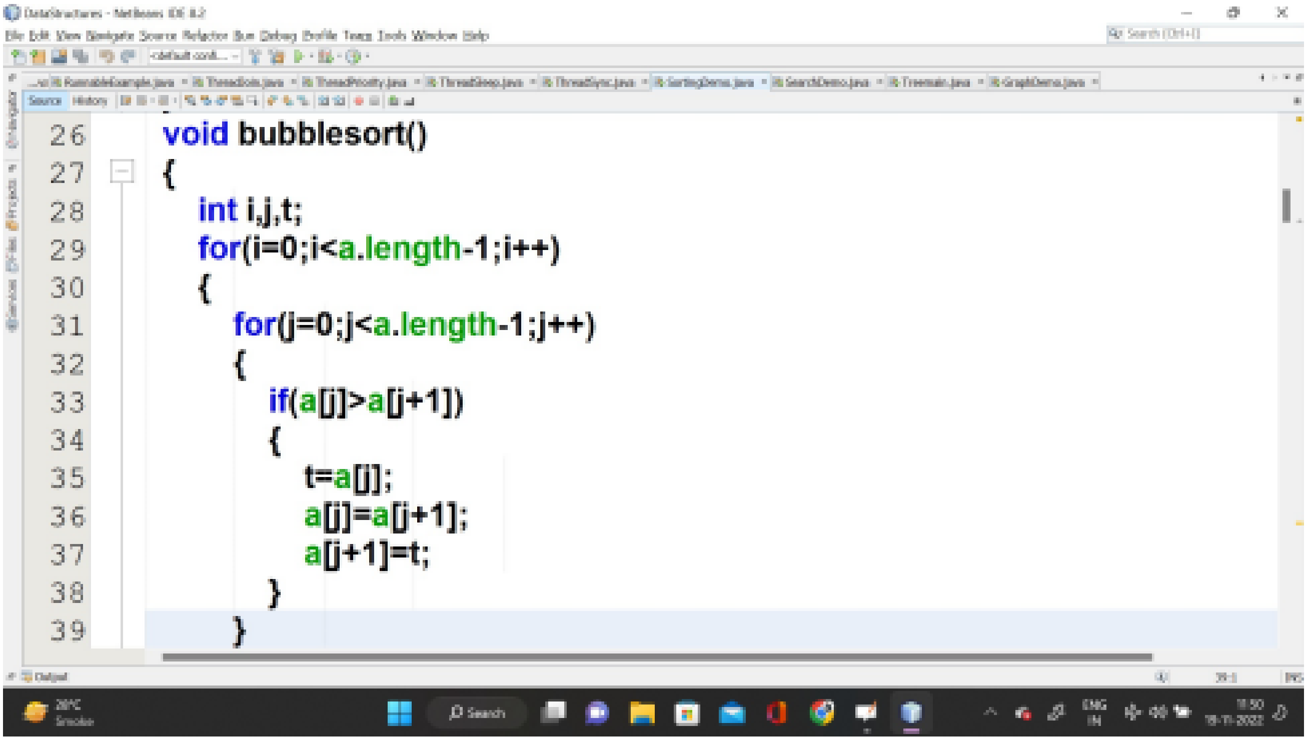
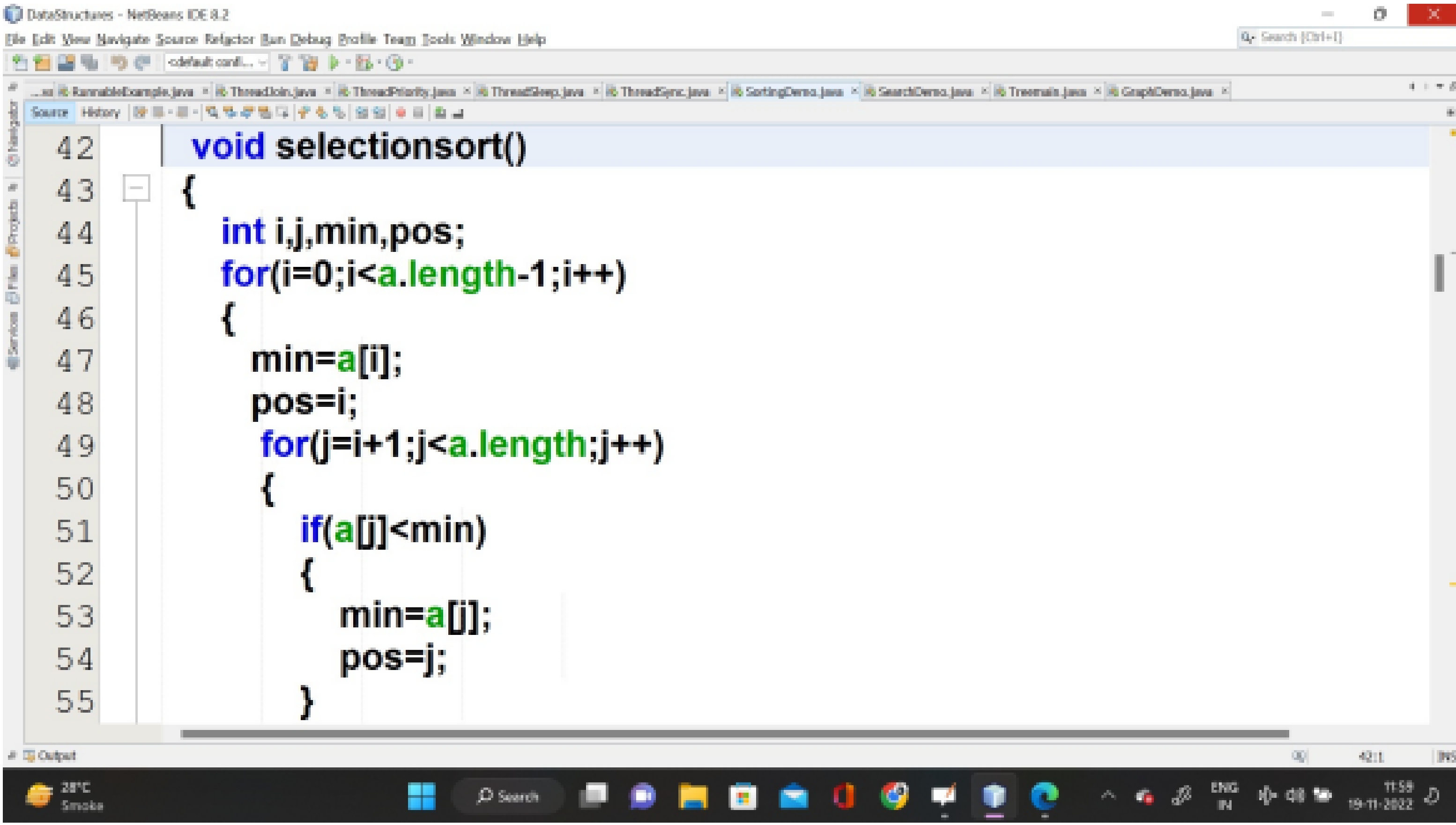


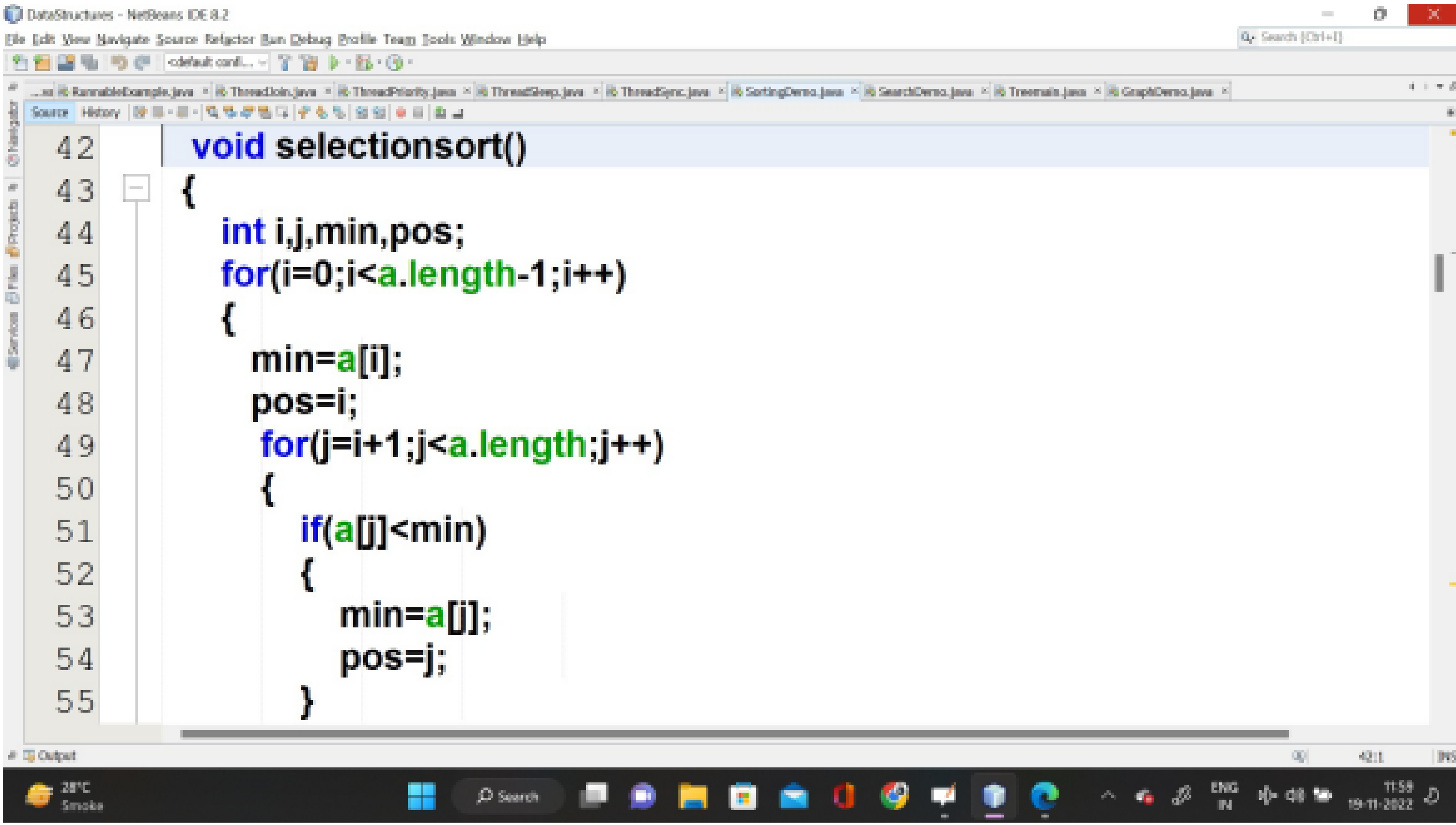
```
void bubblesort()
{
    int i,j,t;
    for(i=0;i<a.length-1;i++)
    {
        for(j=0;j<a.length-1;j++)
        {
            if(a[j]>a[j+1])
            {
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
```



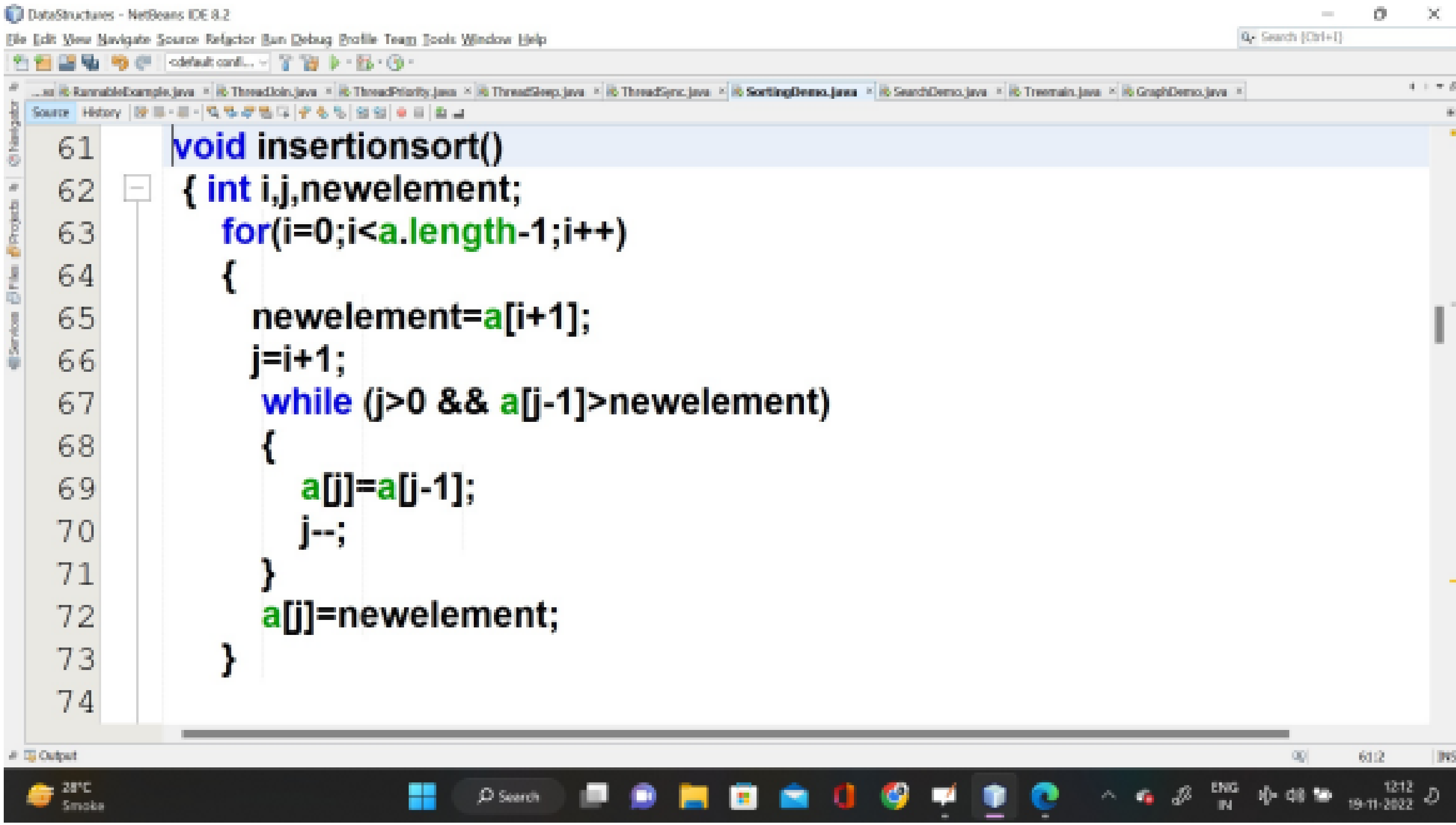
```java
void bubblesort()
{
    int i,j,t;
    for(i=0;i<a.length-1;i++)
    {
        for(j=0;j<a.length-1;j++)
        {
            if(a[j]>a[j+1])
            {
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
```



```
void selectionsort()
{
   int i,j,min,pos;
   for(i=0;i<a.length-1;i++)
   {
     min=a[i];
     pos=i;
      for(j=i+1;j<a.length;j++)
      {
         if(a[j]<min)
         {
            min=a[j];
            pos=j;
         }
```

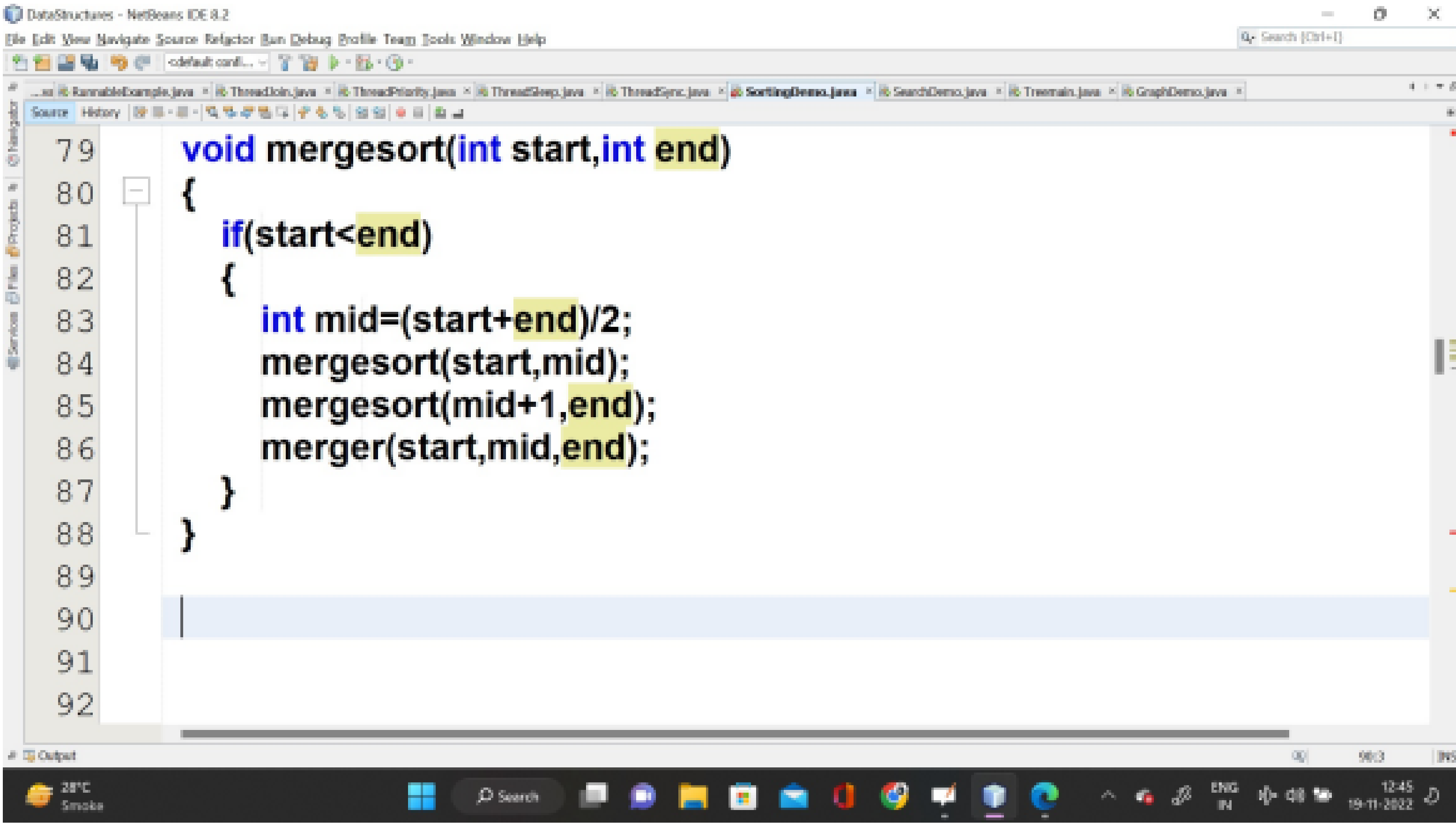


```
 void selectionsort()
{
    int i,j,min,pos;
    for(i=0;i<a.length-1;i++)
    {
       min=a[i];
       pos=i;
        for(j=i+1;j<a.length;j++)
        {
            if(a[j]<min)
            {
                min=a[j];
                pos=j;
            }
```



```
void insertionsort()
 { int i,j,newelement;
    for(i=0;i<a.length-1;i++)
    {
      newelement=a[i+1];
      j=i+1;
       while (j>0 && a[j-1]>newelement)
       {
          a[j]=a[j-1];
          j--;
       }
       a[j]=newelement;
    }

```



```
void mergesort(int start,int end)
{
    if(start<end)
    {
        int mid=(start+end)/2;
        mergesort(start,mid);
        mergesort(mid+1,end);
        merger(start,mid,end);
    }
}
```

```
void merger(int start,int mid,int
  end) {
     int temp[]=new
     int[a.length]; int i,j,index;
     i=index=start;
    j=mid+1;
  while(i<=mid && j<=end)
  {
     if(a[i]<a[j])
        temp[index++]=a[i++];
     else
        temp[index++]=a[j++];
  }
  while(i<=mid)
     temp[index++]=a[i++];
  while(j<=end)
     temp[index++]=a[j++];

  for (i=start;i<=end;i++)
     a[i]=temp[i];

}
 void quicksort(int start,int end)
  {
     int i,j,pivot;
     i=start; j=end;pivot=start;
     while(i<j)
     {
        while(a[i]<a[pivot])
           i++;
        while(a[j]>a[pivot])
            j--;
        if(i<j)
        {
            int t=a[i];
            a[i]=a[j];
            a[j]=t;
        }
    }
    if(i<end)
        quicksort(i+1,end);
    if(start<j)
        quicksort(start,j-1);
}
      void heap()
       {
       int i,k,lp,temp,done=0;
       for(i=a.length-1;i>=0;i--) {
       for(k=0;k<=i;k++)
       {
       done=0;
       lp=k;
       while(lp>0 && done!=1) {
       if(a[lp]>a[lp/2])
       {
```

```
temp=a[lp];
a[lp]=a[lp/2];
a[lp/2]=temp;
lp=lp/2;
}
else
done=1;
}
}
temp=a[0];
a[0]=a[i];
a[i]=temp;
}
}
        void binarysearch(int start,int end,int key)
          {
             if(start<=end)
             {
               int mid=(start+end)/2;
               if(a[mid]==key)
                  System.out.println("found at:"+(mid+1));

               if(key<a[mid])//left only
                       binarysearch(start,mid-1,key);

               if(key>a[mid])//right only
```

```
         binarysearch(mid+1,end,key);
  }
  else
     System.out.println("Not found");
}
```