```
Merge_sort(A,lb,ub)
{
        If(lb < ub)
        {
                mid = (up+lb)/2
                Merge_sort(A,lb,mid)
                Merge_sort(A,mid+1,ub)
                Merge(A, lb, mid, ub)

        }
}

Merge(A, lb, mid, ub)
{
         i=lb;j=mid+1;k=lb;
         while(i<= mid && j <=
         ub)
                {
                        if(A[i] <= A[j])
                        {
                                B[k]=A[i];
                                i++; k++;
                        }
                        else
                            {
                                B[k]=A[j
                                ]; j++;
                                k++;
                }               }

        while( i <= mid)
          {
              B[k]=A[i];
              i++; k++;
           }
          while( j <= ub)
          {
              B[k]=A[j]; j++;
              k++;
           }
         for(i=lb; i<=ub;i++)
          {A[k]=B[i]; }
 }
```

```
quicksort ( A , lb , ub )
{

  If ( lb < ub )

    {
      q = Partition ( A , lb ,
      ub ); quicksort ( A , lb
      , q-1 ); quicksort ( A ,
      q+1 , ub );
    }
}
 Partition ( A , lb , ub )
{
  pivot = A[lb];
  start = lb+1;
  end = ub;
  while ( start < end )
    {
        while ( A[start] <= pivot )
          {
            start ++;
          }
        while ( A[end] > pivot )
        {
           end - -;
        }
        if ( start < end )
         {
            swap( A[srart] , A[end] );
         }
      }
   swap( A[lb] ,
   A[end] ); return
   end;
}
```