

```
# GAP code implementing some sorting algorithms
# Try executing BubbleSort(list1) or MergeSort(list1)...
# There is redundant stuff in the code so that the algorithm prints some
# output allowing you to follow the algorithms!

list1 := [3,2,1,3,5,6,3,2,4,5,7,7,8,9,4,2,1,1,3,5,6,7,83,2,5];
list2 := [25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1];

# This is the standard implementation of bubble sort...

BubbleSort := function(list) local i, j, n, temp;
  n := Length(list);
  for i in [1..n-1] do
    for j in Reversed([i+1..n]) do
      Print(j-1, " ");
      if list[j-1] > list[j] then
        temp := list[j]; list[j] := list[j-1]; list[j-1] := temp;
      fi;
    od;
    Print("\n");
  od;
  Print("\n");
  return list;
end;

# This is a slightly sloppy implementation of merge sort
# First we need a subroutine to take two lists of numbers in ascending order
# and merge them together to produce a single list in ascending order.

MergeLists := function(list1,list2,x,y) local list,i,j;
  i := 1;
  j := 1;
  list := [];
  while IsBound(list1[i]) and IsBound(list2[j]) do
    Print("(" , x+i-1, ", ", y+j-1, ") ");
    if list1[i] < list2[j] then
      Add(list,list1[i]); i := i+1;
    else
      Add(list,list2[j]); j := j+1;
    fi;
  od;
  if IsBound(list1[i]) then
    while IsBound(list1[i]) do
      Add(list,list1[i]);
      i := i+1;
    od;
  fi;
end;
```

```

else
  while IsBound(list2[j]) do
    Add(list,list2[j]);
    j := j+1;
  od;
fi;
return list;
end;

# Here is the main merge sort algorithm

MergeSort := function(list) local n,cuts,i,jump,new;
n := Length(list);
cuts := [];
for i in [1..n] do Add(cuts,[list[i]]);od;
jump := 1;
while jump < n do
  Print(cuts,"\n");
  i := 1;
  while i + jump <= n do
    Print("Merging comparisons: ");
    new := MergeLists(cuts[i], cuts[i + jump], i, i + jump);
    Print("\n");
    cuts[i] := new;
    Unbind(cuts[i + jump]);
    i := i + 2 * jump;
  od;
  jump := 2 * jump;
od;
Print("\n");
return cuts[1];
end;

```