# Merge Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAXSIZE 1000
int A[MAXSIZE];
/* Function prototypes */
void mergeSort ( int, int );
void merge ( int, int, int );
void printArray ( int );
```

# Merge Sort

```
void mergeSort ( int i , int j )
/* i and j are the leftmost and rightmost indices
of the current part of the array being sorted. */
{
int mid;
if (i == j) return; /* Array of size 1 */
mid = (i + j) / 2; /* Compute mid index */
mergeSort(i,mid); /* Sort the left half */
mergeSort(mid+1,j); /* Sort the right half */
merge(i,mid,j);
/* Merge the two sorted subarrays */
}
```

# Merge Sort

```
void merge ( int i1, int j1, int j2 ) {
int i2, k1, k2, k;
int tmpArray[MAXSIZE];
i2 = j1 + 1;
k1 = i1; k2 = i2; k = 0;
while ((k1 <= j1) || (k2 <= j2)) {
if (k1 > j1) { /* Left half is exhausted */
/* Copy from the right half */
tmpArray[k] = A[k2];
++k2;
} else if (k2 > j2) { /* Right half is exhausted */
/* Copy from the left half */
tmpArray[k] = A[k1];
++k1;
}
```

# Merge Sort

```
 else if (A[k1] < A[k2]) {
 /* Left pointer points to a smaller value */
/* Copy from the left half */
tmpArray[k] = A[k1]; ++k1;
} else { /* Right pointer points to a smaller value
/* Copy from the right half */
tmpArray[k] = A[k2]; ++k2;
}
++k; /* Advance pointer for writing */
}/* while end*/
/* Copy temporary array back to the original array
--k;
while (k >= 0) {
A[i1+k] = tmpArray[k];
--k; }
```

# Merge Sort

```
void printArray ( int s ) {
int i;
for (i=0; i<s; ++i) printf("%d ",A[i]);
}
int main () {
int s, i; scanf("%d",&s);
for (i=0; i<s; ++i) A[i] = 1 + rand() % 99;
printf("Array before sorting:"); printArray(s);
mergeSort(0,s-1);
printf("Array after sorting:"); printArray(s);
}
```