```
# include<stdio.h>
# include <stdlib.h>
# include<sys/types.h>
# include<unistd.h>
int split ( int[], int , int );
void quickSort(int* ,int, int);
void mergeSort(int arr[],int low,int mid,int high)
{
int i,j,k,l,b[20];
l=low;
i=low;
j=mid+1;
while((l<=mid)&&(j<=high)){
if(arr[l]<=arr[j]){
b[i]=arr[l];
l++;
}
else{
b[i]=arr[j];
j++;
}
i++;
}
if(l>mid){
for(k=j;k<=high;k++){
b[i]=arr[k];
i++;
}
}
else{
for(k=l;k<=mid;k++){
```

```
b[i]=arr[k];
i++;}
}
for(k=low;k<=high;k++)
{
arr[k]=b[k];
}
}
void partition(int arr[],int low,int high)
{
int mid;
if(low<high)
{
double temp;
mid=(low+high)/2;
partition(arr,low,mid);
partition(arr,mid+1,high);
mergeSort(arr,low,mid,high);
}
}
void display(int a[],int size){
int i;
for(i=0;i<size;i++){
printf("%d\t\t",a[i]);
}
printf("\n");
}
int main()
{
int pid, child_pid;
int size,i,status;/* Input the Integers to be sorted */
```

```
printf("Enter the number of Integers to Sort::::\t");
scanf("%d",&size);
int a[size];
int pArr[size];
int cArr[size];
for(i=0;i<size;i++){
printf("Enter number %d:",(i+1));
scanf("%d",&a[i]);
pArr[i]=a[i];
cArr[i]=a[i];
}
/* Display the Enterd Integers */
printf("Your Entered Integers for Sorting\n");
display(a,size);
/* Process ID of the Parent */
pid=getpid();
printf("Current Process ID is : %d\n",pid);
/* Child Process Creation */
printf("[ Forking Child Process ... ] \n");
child_pid=fork();
/* This will Create Child Process and
Returns Child's PID */
if( child_pid < 0){/* Process Creation Failed ... */
printf("\nChild Process Creation Failed!!!!\n");
exit(-1);
}
else if( child_pid==0) {
/* Child Process */
printf("\nThe Child Process\n");
printf("\nchild process is %d",getpid());
printf("\nparent of child process is %d",getppid());
```

```
printf("Child is sorting the list of Integers by QUICK SORT::\n");
quickSort(cArr,0,size-1);
printf("The sorted List by Child::\n");
display(cArr,size);
printf("Child Process Completed ...\n");
sleep(10);
printf("\nparent of child process is %d",getppid());
}
else {
/* Parent Process */
printf("parent process %d started\n",getpid());
printf("Parent of parent is %d\n",getppid());
sleep(30);
printf("The Parent Process\n");
printf("Parent %d is sorting the list of Integers by MERGE SORT\n",pid);
partition(pArr,0,size-1);
printf("The sorted List by Parent::\n");
display(pArr,size);
// wait(&status);
printf("Parent Process Completed ...\n");
}
return 0;
}int split ( int a[ ], int lower, int upper )
{
int i, p, q, t ;
p = lower + 1 ;
q = upper ;
i = a[lower] ;
while ( q >= p )
{
while ( a[p] < i )
```

```
p++ ;
while ( a[q] > i )
q-- ;
if ( q > p )
{
t = a[p] ;
a[p] = a[q] ;
a[q] = t ;
}
}
t = a[lower] ;
a[lower] = a[q] ;
a[q] = t ;
return q ;
}
void quickSort(int a[],int lower, int upper){
int i ;
if ( upper > lower )
{
i = split ( a, lower, upper ) ;quickSort ( a, lower, i - 1 ) ;
quickSort ( a, i + 1, upper ) ;
}
}
```