TE IT

Name : Omkar Gurav

Roll no : 8048

Implement the C program in which main program accepts the integers to be sorted. Main program uses the FORK system call to create a new process called a child process. Parent process sorts the integers using Merge sort algorithm and waits for child process using WAIT system call to sort the integers using Quick sort algorithm. Also demonstrate Zombie and Orphan state.

```
#include <stdio.h>
#include <unistd.h>

void swap(int *x,int *y)
{
 *x=*x+*y;
 *y=*x-*y;
 *x=*x-*y;
}

int qsort(int arr[],int f,int l)
{
 int pivot=arr[l],i=f-1,k;

 for(int j=f;j<=l;j++)
 {
  if(arr[j]<pivot)
  {
   i++;
```

```
  k=arr[i];
  arr[i]=arr[j];
  arr[j]=k;
 }
}
k=arr[i+1];
arr[i+1]=arr[l];
arr[l]=k;

return (i+1);
}

void quicksort(int arr[],int f,int l)
{
 if(f<l)
 {
  int p=qsort(arr,f,l);
  quicksort(arr,f,p-1);
  quicksort(arr,p+1,l);
 }
}

void mergesort(int list[],int low,int mid,int high)
{
 int i=low,mi=mid+1,lo=low,temp[50];

 while(lo<=mid && mi<=high)
 {
  if(list[lo]<=list[mi])
  {
```

```
 temp[i]=list[lo];
 lo++;
}

else
{
 temp[i]=list[mi];
 mi++;
}

 i++;
}

if(lo>mid)
for(int k=mi;k<=high;k++)
{
 temp[i]=list[k];
 i++;
}

else
 for(int k=lo;k<=mid;k++)
 {
  temp[i]=list[k];
  i++;
 }

 for(int k=low;k<=high;k++)
 list[k]=temp[k];
}
```

```
void partition(int arr[],int l,int h)
{
 if(l<h)
 {
  int m=(l+h)/2;
  partition(arr,l,m);
  partition(arr,m+1,h);
  mergesort(arr,l,m,h);
 }
}

void display(int arr[],int n)
{
 for(int i=0;i<n;i++)
 printf("%d\t",arr[i]);

 printf("\n");
}


int main()
{
 pid_t pid;
 int num[20],count;

 printf("\nEnter no. of integers to be sorted: ");
 scanf("%d",&count);

 printf("\nEnter integers\n");
```

```
for(int i=0;i<count;i++)
scanf("%d",&num[i]);

 pid=fork();

 if(pid>0)
 {
  wait(NULL);
  printf("\n\nInside parent process");
  printf("\nProcess ID:%d",getpid());
  printf("\n\n");

  quicksort(num,0,count-1);

  printf("\nIntegers sorted using Quicksort\n");
  display(num,count);
  printf("\n");
}

else if(pid==0)
{
printf("\n\nInside child process");
printf("\nProcess ID:%d",getpid());
printf("\n\n");

partition(num,0,count-1);

printf("\nIntegers sorted using Mergesort\n");
display(num,count);
printf("\n");
```

```
 }

 else
 printf("\nChild process could not be created!\n");

 return 0;
}
```

Output:

```
Enter no. of integers to be sorted: 6

Enter integers
2
7
4
1
9
6

Inside child process
Process ID:3090

Integers sorted using Mergesort
1     2     4     6     7     9
```

Inside parent process

Process ID:3089

Integers sorted using Quicksort

1 2 4 6 7 9

Zombie process program :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
 pid_t pid;
pid=fork();

if(pid>0)
{
 sleep(10);
 printf("\nIn parent process\n");
}

else
```

```
 {
  printf("\nIn child process\n");
  exit(0);
 }

 return 0;
}
```

Output:

In child process

In parent process

Orphan process program :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
 pid_t pid;

 pid=fork();
```

```
 if(pid>0)
{
 printf("\nIn parent process\n");
}

 else
 {
  sleep(10);
  printf("\nIn child process\n");
 }

 return 0;
}
```

Output:

In parent process