Name: Aditya Somani
Roll No: T1851061
Div: A
PRN NO. 71901204L

ASSIGNMENT NO. 2a

TITLE: Process control system calls: The demonstration of FORK, EXECVE and WAIT system calls along with zombie and orphan states.

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
void quickSort(int [],int ,int );
int partition(int [],int ,int );
void mergeSort(int [],int ,int );
void merge(int [],int ,int ,int ,int );
int main()
{
	int i,j,n;
	pid_t pid;
	int arr[30];
	printf("\nEnter the number of elements:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&arr[i]);
	}

	pid=fork();
	if(pid<0)
	{
		printf("Error occured:");
	}
	else if(pid==0)
	{
		printf("\n\t ****This is child process**** ");
		printf("\n\t My process id is : %d %d", getpid(),getppid());

		quickSort(arr,0,n-1);
		printf("\nQuicksort");
		for(i=0;i<n;i++)
		printf(" %d",arr[i]);
		printf("\n\n");
	}
	else
	{
		//wait();

		printf("\n***Parent process resumed after the execution of child process with
PID***");
		printf("\n\t My process id is : %d", getpid());
		printf("\n\t My Parent process id is : %d", getppid());
		mergeSort(arr,0,n-1);
		printf("\nMergesort:");
```

```
f                   0;i<n;i++) printf("
o                   %d",arr[i]);
r                   printf("\n\n");
(
i
=

          }

        return 0;
```

```
}
void quickSort(int arr[],int low,int high)
{
	int j;
	if(low<high)
	{
		j=partition(arr,low,high);
		quickSort(arr,low,j-1);
		quickSort(arr,j+1,high);
	}
}
int partition(int arr[],int low,int high)
{
	int i,j,temp,pivot;
	pivot=arr[low];
	i=low;
	j=high+1;
	do
	{
		do
		i++;
		while(arr[i]<pivot && i<=high);
		do
		j--;
		while(arr[j]>pivot);
		if(i<j)
		{
			temp=arr[i];
			arr[i]=arr[j];
			arr[j]=temp;
		}
	}
	while(i<j);
	arr[low]=arr[j];
	arr[j]=pivot;
	return(j);
}
void mergeSort(int arr[],int low,int high)
{
	int mid;
	if(low<high)
	{
		mid=(low+high)/2;
		mergeSort(arr,low,mid);
		mergeSort(arr,mid+1,high);
		merge(arr,low,mid,mid+1,high);
	}
}
void merge(int arr[],int i1,int j1,int i2,int j2)
{
	int temp[50];
	int i,j,k;
```

```
        i=i1;
        j=i2;
        k=0;
        while(i<=j1 && j<=j2)
        {
                if(arr[i]<arr[j])
                temp[k++]=arr[i++];
                else
                temp[k++]=arr[j++];
        }
        while(i<=j1)
    temp[k++]=arr[i++];
        while(j<=j2)
        temp[k++]=arr[j++];
        for(i=i1,j=0;i<=j2;i++,j++)
        arr[i]=temp[j];
}
```