// 2

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

//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 sorting algorithm and waits for child process using WAIT system call to

//sort the integers using any sorting algorithm. Also demonstrate zombie and orphan states.

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

void bass(int arr[30],int n)

{

int i,j,temp;

for(i=0;i<n;i++)

{

for(j=0;j<n-1;j++)

{

if(arr[j]>arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("\n Ascending Order \n");

for(i=0;i<n;i++)

printf("\t%d",arr[i]);

printf("\n\n\n");

}

void bdsc(int arr[30],int n)

{

int i,j,temp;

for(i=0;i<n;i++)

{

for(j=0;j<n-1;j++)

{

if(arr[j]<arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("\n Descending Sorting \n\n");

for(i=0;i<n;i++)

printf("\t%d",arr[i]);

printf("\n\n\n");

}

void forkeg()

{

int arr[25],arr1[25],n,i,status;

printf("\nEnter the no of values in array");

scanf("%d",&n);

printf("\nEnter the array elements");

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

int pid;

pid=fork();

if(pid==0)

{

sleep(10);

printf("\nchild process\n");

printf("child process id=%d\n",getpid());

bdsc(arr,n);

printf("\nElements Sorted Using Quick Sort");

printf("\n");

for(i=0;i<n;i++)

printf("%d,",arr[i]);

printf("\b");

printf("\nparent process id=%d\n",getpid());

system("ps -x");

}

else

{

printf("\nparent process\n");

printf("\nparent process id=%d\n",getpid());

bass(arr,n);

printf("Elements Sorted Using Bubble Sort");

printf("\n");

for(i=0;i<n;i++)

printf("%d,",arr[i]);

printf("\n\n\n");

}

}

int main()

{

forkeg();

return 0;

}