2A

```
/*Problem Statement: 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>


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=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",getppid());
```

```
                    system("ps -x");
      }
     else
      {
                    printf("\nparent process\n");
                    printf("\nparent process id=%d\n",getppid());
                    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;
 }


/*


parth@parth-ThinkCentre-E73:~$ gcc ass2a.c
parth@parth-ThinkCentre-E73:~$ ./a.out
```

```
Enter the no of values in array5


Enter the array elements5 3 6 2 6 2


parent process


parent process id=2797


 Ascending Order
        2       3       5       6       6


Elements Sorted Using Bubble Sort
2,3,5,6,6,
parth@parth-ThinkCentre-E73:~$
child process
child process id=2816


 Descending Sorting


        6       6       5       3       2
Elements Sorted Using Quick Sort
6,6,5,3,2,
parent process id=1798


*/
```

2B

/*Problem Statement: 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>

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 Soring \n");

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

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


}


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");
```

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


}
void forkeg()
{
          int arr[25],arr1[25],n,i,status;
          printf("\nEnter Size : - \n");
          scanf("%d",&n);
          printf("\nEnter Array : - \n");
          for(i=0;i<n;i++)
                    scanf("%d",&arr[i]);
          int pid=fork();
          if(pid==0)
          {
                    printf("\nChild Id = %d\n",getpid());
                    bdsc(arr,n);
                    printf("\n Bubble Sorting \n");
                    printf("\n");
                    for(i=0;i<n;i++)
                              printf("\t %d,",arr[i]);
                    printf("\n");
          }
          else
```

```
            {
                        sleep(10);


                        printf("Parent Id = %d\n",getppid());
                        bass(arr,n);
                        printf("\n Bubble Sort : \n");
                        printf("\n");
                        for(i=0;i<n;i++)
                                  printf("%d,",arr[i]);
                                  printf("\b");
                        system("ps -x");
       }
 }


int main()
 {
   forkeg();
   return 0;
 }


/*
parth@parth-ThinkCentre-E73:~$ gcc ass2b.c
parth@parth-ThinkCentre-E73:~$ ./a.out


Enter Size : -
```

```
5


Enter Array

: - 3 4 2 9 1


Child Id = 2866


 Descending Sorting

              9          4          3          2          1




 Bubble Sorting



               9,         4,         3,         2,         1,

Parent Id = 2797


 Ascending Soring

              1          2          3          4          9

 Bubble Sort :


*/
```