```
/*
	Name: Mihir Kulkarni
	Roll no. : 33132 L9
	Assignment 2(a)
*/

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

/*void bubble_sort(int arr[],int size)
{
	int temp;
	for(int i=0;i<size-1;i++)
	{
		for(int j=0;j<size-i-1;j++)
		{
			if(arr[j]>arr[j+1])
			{

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

	//printf("\nSorted elements:-\n");
   //for(int i=0;i<size;i++)
   //	printf("%d ",arr[i] );
}*/
void swap(int* a, int* b)
{
   int t = *a;
   *a = *b;
   *b = t;
}
int partition (int arr[], int low, int high)
{
   int pivot = arr[high]; // pivot
   int i = (low - 1); // Index of smaller element

   for (int j = low; j <= high - 1; j++)
   {
      // If current element is smaller than the pivot
      if (arr[j] < pivot)
      {
         i++; // increment index of smaller element
         swap(&arr[i], &arr[j]);
      }
   }
```

```
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[p] is now
        at right place */
        int pi = partition(arr, low, high);

        // Separately sort elements before partition and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
       are any */
    while (i < n1) {
```

```
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
       are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
            pid_t pid;
            int size,choice;
            printf("\nEnter the size array :");
            scanf("%d",&size);
            int arr[size];
            printf("\nEnter the array elements :");
            for(int i=0;i<size;i++)
                        scanf("%d",&arr[i]);

    printf("Enter process which you want to execute:-\n1.Orphan\n2.Zombie\n=>");
    scanf("%d",&choice);

    switch(choice)
    {
            case 1:
                        pid = fork();
                        if(pid==-1)
                        {
                                    printf("\nERROR");
```

```
        }
        if(pid==0)
        {

                printf("\nChild process sleeping...");
                printf("\nChild process : %d",getpid());

                sleep(5);

                printf("\nChild process executing...");
                printf("\nChild process id: %d",getpid());
                printf("\nParent process id: %d",getppid());

                mergeSort(arr, 0, size - 1);
                printf("\nSorted elements:-\n");
                for(int i=0;i<size;i++)
                        printf("%d ",arr[i] );
                printf("\n");
                printf("\nParent process id: %d",getppid());
                system("ps -elf|grep a.out");
        }
        else
        {
                system("wait");
                printf("\nParent process executing...");
                printf("\nChild process id: %d",getpid());
                printf("\nParent process id: %d",getppid());

                        mergeSort(arr, 0, size - 1);
                printf("\nSorted elements:-");
                for(int i=0;i<size;i++)
                        printf("%d ",arr[i] );
                printf("\n");

        }
        break;
case 2:

        printf("Main process id:%d\n",getpid());
        pid = fork();
        if(pid==-1)
        {
                printf("\nERROR");
        }
        if(pid==0)
        {
                system("wait");
                printf("\nChild process id = %d\n",getpid());
                        quickSort(arr, 0, size - 1);
                        printf("\nSorted elements:-");
                for(int i=0;i<size;i++)
                        printf("%d ",arr[i] );
```

```
                printf("\n");
            }
            else
            {
                sleep(5);
                printf("Parent Id = %d\n",getppid());
                quickSort(arr, 0, size - 1);
                for(int i=0;i<size;i++)
                    printf("%d ",arr[i] );
                printf("\n");
                system("ps -elf|grep a.out");
            }

            break;

        default:
            printf("\nInvalid entry");
    }

        return 0;
}
```

```
mihir@pop-os:~/TE/OS-lab/a2-1$ gcc a2-1.c
mihir@pop-os:~/TE/OS-lab/a2-1$ ./a.out

Enter the size array :5

Enter the array elements :3 5 2 4 1
Enter process which you want to execute:-
1.Orphan
2.Zombie
=>1

Child process sleeping...

Parent process executing...
Child process id: 58193
Parent process id: 57890
Sorted elements:-1 2 3 4 5
mihir@pop-os:~/TE/OS-lab/a2-1$ Child process : 58194
Child process executing...
Child process id: 58194
Parent process id: 1751
Sorted elements:-
1 2 3 4 5

1 S mihir      58194    1751  0  80   0 -   622 do_wai 09:12 pts/3    00:00:00 ./a.out
0 S mihir      58197   58194  0  80   0 -   652 do_wai 09:12 pts/3    00:00:00 sh -c ps -elf|grep a.out
0 S mihir      58199   58197  0  80   0 -  4699 pipe_w 09:12 pts/3    00:00:00 grep a.out
Parent process id: 1751
mihir@pop-os:~/TE/OS-lab/a2-1$ ./a.out

Enter the size array :5

Enter the array elements :3 5 2 4 1
Enter process which you want to execute:-
1.Orphan
2.Zombie
=>2
Main process id:58291

Child process id = 58304

Sorted elements:-1 2 3 4 5
Parent Id = 57890
1 2 3 4 5
0 S mihir      58291   57890  0  80   0 -   622 do_wai 09:13 pts/3    00:00:00 ./a.out
1 Z mihir      58304   58291  0  80   0 -     0 -      09:13 pts/3    00:00:00 [a.out] <defunct>
0 S mihir      58306   58291  0  80   0 -   652 do_wai 09:14 pts/3    00:00:00 sh -c ps -elf|grep a.out
0 S mihir      58308   58306  0  80   0 -  4699 pipe_w 09:14 pts/3    00:00:00 grep a.out
mihir@pop-os:~/TE/OS-lab/a2-1$
```