```c
#include <stdio.h>
void merge(int arr[], int beg, int mid, int end)
{
        int n1 = mid - beg;
        int n2 = end - mid +1;
        int left[n1], right[n2];

        int k = beg;

        for(int i=0;i<n1;i++)
                left[i] = arr[k++];
        for(int i=0;i<n2;i++)
                right[i] = arr[k++];


        k = beg;
        int i = 0;
        int j = 0;


        while(i < n1 && j < n2)
        {
                if(left[i] < right[j])
                {
                        arr[k] = left[i++];
                }
                else
                {
                        arr[k] = right[j++];
                }
                k++;
        }

        while(i < n1)
        {
                arr[k++] = left[i++];
        }
        while(j < n2)
        {
                arr[k++] = right[j++];
        }
}

void mergeSort(int arr[], int beg, int end)
{
        if(beg < end)
        {
                int mid = (beg + end) / 2;

                mergeSort(arr, beg, mid);
                mergeSort(arr, mid +1, end);

                merge(arr, beg, mid+1, end);

                sleep(1);
        }
}

void bubbleSort(int arr[], int n)
{
        for(int i=0;i<n-1;i++)
        {
                for(int j=0;j<n-i-1;j++)
                {
                        if(arr[j] > arr[j + 1])
                        {
                                int tmp = arr[j];
                                arr[j] = arr[j+1];
                                arr[j+1] = tmp;
                        }
                }
                sleep(1);
        }
}

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

int main()
{
        printf("\nMain Parent: %d\n", getppid());

        int p_n = 10;
        int c_n = 4;
        int p_arr[10] = {9,6,3,7,4,1,8,5,2,10};
        int c_arr[4] = {4,3,1,2};

        int id = fork();

        if(id == 0)
        {
                printf("\nChild: Starting\n");
                printf("\nChild: Parent ID: %d\n", getppid());
                printf("\nChild: My ID: %d\n", getpid());
                printf("\nChild: Started Sorting\n");

                bubbleSort(c_arr, c_n);

                printf("\nChild: Sorting Complete\n");
                printarr(c_arr, c_n);

                printf("\nChild: Became Zombie\n");
        }
        else
        {
                printf("\nParent: Starting\n");
                printf("\nParent: Parent ID: %d\n", getppid());
                printf("\nParent: My ID: %d\n", getpid());
                printf("\nParent: Started Sorting\n");

                mergeSort(p_arr, 0, p_n -1);

                printf("\nParent: Sorting Complete\n");
                printarr(p_arr, p_n);

                wait(NULL);

                printf("\nParent: Exiting\n");

        }
        return 0;
}

/*OUTPUT-
Main Parent: 2691

Parent: Starting

Parent: Parent ID: 2691

Parent: My ID: 7640

Parent: Started Sorting

Child: Starting

Child: Parent ID: 7640

Child: My ID: 7641

Child: Started Sorting

Child: Sorting Complete
1 2 3 4

Child: Became Zombie

Parent: Sorting Complete
1 2 3 4 5 6 7 8 9 10

Parent: Exiting

*/
/*OUTPUT TOP-
Tasks: 168 total,   1 running, 131 sleeping,   0 stopped,   1 zombie
*/
```