```c
//PARENT
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

void toString(char *args[], char *cmd, int arr[], int n)
{
    args[0] = cmd;
    for(int i = 1;i<=n;i++)
    {
        char *buffer = malloc(sizeof(char) * 10);
        sprintf(buffer, "%d", arr[i-1]);
        args[i] = buffer;
    }
    args[n+1] = NULL;
}

int main()
{
    printf("PARENT : STARTED\n");
    int arr[10];
    int n = 10;
    printf("PARENT : Enter 10 elements: ");
    for(int i=0;i<n;i++)
    {
        scanf("%d", &arr[i]);
    }

    for(int i=0;i<n-1;i++)
    {
        for(int j = 0;j<n-i-1;j++)
        {
            if(arr[j] > arr[j+1])
            {
                swap(&arr[j], &arr[j+1]);
            }
        }
    }

    printf("PARENT : SORTING COMPLETE\n");

    printf("PARENT : SORTED ELEMENTS : ");
    for(int i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");

    char cmd[] = "./children";
    char *args[n + 2];
    char *envp[] = {NULL};

    toString(args, cmd, arr, n);

    printf("PARENT : Calling Children...\n");

    int pid = fork();
    if(pid == 0)
    {
        if(execve(cmd, args, envp) == -1)
        {
            printf("PARENT : FAILED EXECVE\n");
        }
    }

    wait(NULL);
    printf("PARENT : EXITTING\n");
    return 0;
}

//CHILDREN
#include <stdio.h>
#include <unistd.h>
```

```
int main(int argc, char *argv[])
{
    printf("CHILD : STARTED\n");

    printf("CHILD : RECEIVED ARGUEMENTS\n");
    for (int i = 0; i < argc; i++)
    {
        printf("CHILD : args[%d] : %s\n", i, argv[i]);
    }

    char *cmd = "./children2";
    char *newArgs[argc + 1];
    char *envp[] = {NULL};

    newArgs[0] = cmd;

    int j = 1, n = argc -1;
    for (int i = n; i > 0; i--)
    {
        newArgs[j++] = argv[i];
    }

    newArgs[argc] = NULL;

    if(execve(cmd, newArgs, envp) == -1)
    {
        printf("CHILD : FAILED EXECVE\n");
    }

    return 0;
}

//PRINTING CHILDREN
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("PRINTING CHILD : STARTED\n");

    printf("PRINTING CHILD : REVERSE ORDER IS : ");
    for (int i = 1; i < argc; i++)
    {
        printf("%s ", argv[i]);
    }
    printf("\n");
    return 0;
}

/*OUTPUT-
PARENT : STARTED
PARENT : Enter 10 elements: 8 4 6 2 9 5 1 3 7 10
PARENT : SORTING COMPLETE
PARENT : SORTED ELEMENTS : 1 2 3 4 5 6 7 8 9 10
PARENT : Calling Children...
CHILD : STARTED
CHILD : RECEIVED ARGUEMENTS
CHILD : args[0] : ./children
CHILD : args[1] : 1
CHILD : args[2] : 2
CHILD : args[3] : 3
CHILD : args[4] : 4
CHILD : args[5] : 5
CHILD : args[6] : 6
CHILD : args[7] : 7
CHILD : args[8] : 8
CHILD : args[9] : 9
CHILD : args[10] : 10
PRINTING CHILD : STARTED
PRINTING CHILD : REVERSE ORDER IS : 10 9 8 7 6 5 4 3 2 1
PARENT : EXITTING
*/
```