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

void sort_array(int arr[], int n, int ascending) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if ((ascending && arr[j] > arr[j + 1]) ||
                (!ascending && arr[j] < arr[j + 1])) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

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

int main() {
    int n;
```

```c
printf("Enter number of elements: ");
scanf("%d", &n);

int *arr = (int *)malloc(n * sizeof(int));
if (arr == NULL) {
    printf("Memory allocation failed\n");
    return 1;
}

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

pid_t pid = fork();

if (pid < 0) {
    perror("Fork failed");
    free(arr);
    return 1;
}

if (pid == 0) {
    // Child process - ascending sort
    printf("Child Process: Sorting in Ascending Order:\n");
    sort_array(arr, n, 1);
```

```
        print_array(arr, n);
    } else {
        // Parent process - wait for child and then descending sort
        wait(NULL);
        printf("Parent Process: Sorting in Descending Order:\n");
        sort_array(arr, n, 0);
        print_array(arr, n);
    }

    free(arr);
    return 0;
}
```

```
Output:avcoe@avcoe-Vostro-3710:~$ gcc -o fork fork.c
avcoe@avcoe-Vostro-3710:~$ ./fork
Enter number of elements: 5
Enter 5 elements:
12
23
34
45
56
Child Process: Sorting in Ascending Order:
12 23 34 45 56
Parent Process: Sorting in Descending Order:
56 45 34 23 12
```