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

// Function to perform Bubble Sort
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for(i = 0; i < n-1; i++) {
        for(j = 0; j < n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// Function to perform Selection Sort
void selectionSort(int arr[], int n) {
    int i, j, min, temp;
    for(i = 0; i < n-1; i++) {
        min = i;
        for(j = i+1; j < n; j++) {
            if(arr[j] < arr[min])
                min = j;
        }
        temp = arr[min];
        arr[min] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    int n, i;
    printf("Enter number of integers: ");
    scanf("%d", &n);

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

    pid_t pid = fork();

    if(pid < 0) {
        printf("Fork failed!\n");
        exit(1);
    }
    else if(pid == 0) {  // Child process
        printf("\nChild Process (PID: %d)\n", getpid());
        printf("Sorting using Selection Sort...\n");
        selectionSort(arr, n);

        printf("Child Process Sorted Array: ");
        for(i = 0; i < n; i++)
            printf("%d ", arr[i]);
        printf("\n");

        printf("\nChild process completed. Going to sleep for 5 seconds...\n");
        sleep(5); // To demonstrate Zombie state
        printf("Child exiting now.\n");
        exit(0);
    }
    else {  // Parent process
        printf("\nParent Process (PID: %d), Child PID: %d\n", getpid(), pid);
        printf("Sorting using Bubble Sort...\n");
        bubbleSort(arr, n);

        printf("Parent Process Sorted Array: ");
        for(i = 0; i < n; i++)
            printf("%d ", arr[i]);
        printf("\n");

        printf("\nParent sleeping for 10 seconds to observe zombie state...\n");
        sleep(10); // Child will become zombie during this time

        int status;
        wait(&status); // Collect child's exit status
        printf("Parent collected child process using wait().\n");
    }

    return 0;
}
```

```
nano fork_sort.c
gcc fork_sort.c -o fork_sort
./fork_sort
```

input

```
Enter number of integers: 5

Enter 5 integers: 9 3 5 1 7
```

output

```
Parent Process (PID: 2205), Child PID: 2206

Sorting using Bubble Sort...

Parent Process Sorted Array: 1 3 5 7 9


Parent sleeping for 10 seconds to observe zombie state...


Child Process (PID: 2206)

Sorting using Selection Sort...

Child Process Sorted Array: 1 3 5 7 9


Child process completed. Going to sleep for 5 seconds...

Child exiting now.


(After 10 seconds)

Parent collected child process using wait().
```