```
//File1

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

// Function to sort the array
void bubble_sort(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;
         }
}

int main() {
   int n, arr[50], i;
   pid_t pid;
   char *args[60]; // for execve arguments

   printf("PARENT: My process ID is %d\n", getpid());

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

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

bubble_sort(arr, n);

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

pid = fork();

if (pid < 0) {
   perror("Fork failed");
   exit(1);
}

if (pid == 0) {  // Child process
   printf("\nCHILD: My process ID is %d\n", getpid());
   printf("CHILD: My parent process ID is %d\n", getppid());

   args[0] = "./reverse";  // program name

   // Pass sorted array as command-line arguments
   for (i = 0; i < n; i++) {
      char *num = malloc(10);
```

```
            sprintf(num, "%d", arr[i]);
            args[i + 1] = num;
        }
        args[n + 1] = NULL;

        execve("./reverse", args, NULL);
        perror("execve failed");
        exit(1);
    } else {
        wait(NULL);
        printf("\nPARENT: Child has finished displaying reversed array.\n");
        printf("PARENT: Exiting now. (PID: %d)\n", getpid());
    }

    return 0;
}

//File 2

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int arr[50], i, n = argc - 1;

    printf("\nCHILD PROGRAM (reverse.c):\n");
    printf("Process ID: %d | Parent ID: %d\n", getpid(), getppid());
```

```
    if (n <= 0) {
        printf("No array received from parent.\n");
        return 1;
    }

    printf("\nReceived sorted array: ");
    for (i = 1; i <= n; i++) {
        arr[i - 1] = atoi(argv[i]);
        printf("%d ", arr[i - 1]);
    }

    printf("\nReversed array: ");
    for (i = n - 1; i >= 0; i--)
        printf("%d ", arr[i]);

    printf("\n\nCHILD: Task complete. Exiting now.\n");
    return 0;
}
```