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

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;
         }
      }
   }
}

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

   int arr[n];
   printf("Enter %d elements: ", 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) sorting array...\n", getpid());
    bubbleSort(arr, n);

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

    // Convert integers to strings for execve arguments
    char *args[n + 2];
    args[0] = "./reverse";  // name of new program
    for(i = 0; i < n; i++) {
        char *num = malloc(10);
        sprintf(num, "%d", arr[i]);
        args[i+1] = num;
    }
    args[n+1] = NULL;

    printf("\nExecuting reverse program using execve()...\n");
    execve(args[0], args, NULL);

    // If execve fails
    perror("execve failed");
    exit(1);
}

else {  // Parent Process
```

```
        wait(NULL);
        printf("\nParent Process (PID: %d) finished.\n", getpid());
    }

    return 0;
}
```

nano main_sort.c

program 2

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

int main(int argc, char *argv[]) {
    int i;
    printf("\nNew Program Executed Successfully (PID: %d)\n", getpid());
    printf("Array in Reverse Order: ");

    for(i = argc - 1; i > 0; i--) {
        printf("%s ", argv[i]);
    }
    printf("\n");

    return 0;
}
```

nano reverse.c

compile both :gcc main_sort.c -o main_sort

gcc reverse.c -o reverse

./main_sort

input

Enter number of elements: 5

Enter 5 elements: 9 3 7 1 5

output

Child Process (PID: 3287) sorting array...

Sorted Array by Child: 1 3 5 7 9

Executing reverse program using execve()...

New Program Executed Successfully (PID: 3287)

Array in Reverse Order: 9 7 5 3 1

Parent Process (PID: 3286) finished.