Implement the C program in which main program accepts the integers to be sorted Main program uses the fork system call to create a new process called a child process. Parent process sorts the integers using insertion sort and waits for child process using wait system call to sort the integers using selection sort.

C program that demonstrates how to sort integers using both insertion sort and selection sort in separate processes using the fork() system call and wait() function:

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

#define MAX_SIZE 100

void insertion_sort(int *arr, int size);
void selection_sort(int *arr, int size);

int main()
{
   int nums[MAX_SIZE], size, i;

   printf("Enter the number of integers to be sorted: ");
   scanf("%d", &size);

   printf("Enter the integers: ");
   for (i = 0; i < size; i++) {
      scanf("%d", &nums[i]);
   }

   // Create a child process to sort the array using selection sort
   pid_t child_pid = fork();
```

```
    if (child_pid == 0) {
        // This is the child process
        selection_sort(nums, size);
        exit(0);
    } else {
        // This is the parent process
        insertion_sort(nums, size);
        wait(NULL);  // Wait for child process to complete
    }

    printf("Sorted array: ");
    for (i = 0; i < size; i++) {
        printf("%d ", nums[i]);
    }
    printf("\n");

    return 0;
}

// Insertion sort function
void insertion_sort(int *arr, int size)
{
    int i, j, temp;
    for (i = 1; i < size; i++) {
        temp = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > temp) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
    }
}
```

```
// Selection sort function
void selection_sort(int *arr, int size)
{
   int i, j, min, temp;
   for (i = 0; i < size - 1; i++) {
      min = i;
      for (j = i + 1; j < size; j++) {
         if (arr[j] < arr[min]) {
            min = j;
         }
      }
      temp = arr[i];
      arr[i] = arr[min];
      arr[min] = temp;
   }
}
```

This program first reads in the number of integers to be sorted, and then reads in the integers themselves. It then creates a child process using the fork() system call, and the child process sorts the array using selection sort while the parent process sorts the array using insertion sort. The parent process then waits for the child process to complete using the wait() function before printing the sorted array.