Name:- Kumar D.
Roll:- 33143

Assignment 2B

Problem Statement :-
Implement a C program in which main program accepts integers to be sorted and uses the FORK system call to create a new child process. Parent process sorts the integers in ascending order & ~~child~~ passes the sorted array to child process through the command-line arguments of EXECVE system call. The child process program called through the EXECVE call prints the integers in reverse order

Theory
execve() system call
The execve() system call causes the program that is currently being run by the calling process to be replaced with a new program, with newly initialised stack, heap, and data segments. It is used to run an external program/script from inside of some other program/script.
Synopsis :-

```
#include <unistd.h>

int execve(const char *pathname,
           char *const argv[],
           char *const envp[] )
```

'pathname' must be a binary executable or a script starting with a line of the form :- #! interpretor [optional-arg]
eg.- #! bash etc.

'argv' is an array of pointers to strings passed as command-line arguments to the new program
argv[0] should contain the file name of the file being executed.
argv[] array should be terminated by a NULL pointer
i.e. argv[argc] must be NULL

'envp' is an array of strings of conventionally, key=value pairs which are passed as the environment of the new program. It should also be NULL terminated
(external)

The argv[] & envp[] can be accessed by the new (external) program's main function when it is defined as:

```
int main(int argc, char *argv[], char *envp[])
```

execve() does not return on success, and the text, initialized & unitialized data and stack of the calling process are over-written according to the contents of the newly loaded program.

Conclusion
We demonstrated execve() system call over a sorted array of integers