REPORT :

How to run :

make all

By default it generates 12 random numbers.

By default it does clean.

Bonus parts are done in my homework.

Error handling is done in my homework.

No memory leak exists.

My code first takes arguments. Then generates random numbers. Then does cleanup for if there are existing fifos. Then creates fifos. Then creates signal handler.Then forks.

Child 1 opens fifo1 in read only. Checks error handling. Reads numbers and sum them. Then opens fifo2 with write mode. Sleeps 10 second for not sending the sum earlier than the parent process sends the numbers and the command to fifo2. Then writes sum to fifo2. Then exit.

Child 2 first reads numbers then multiply command then it waits for 10 second (actually don't need but pdf says wait) then it reads sum from fifo2. Then does sum+multiply and prints. Then exit.

Parent sends random numbers to fifo1 and fifo2. Then sends multiply command to fifo2. Then controls if child exit counter . Waits if counter is not 2 . Else exits.

- Create a program that takes an integer argument.

```
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <number of elements>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
```

- Create two FIFOs

```
    cleanup();  // Remove any existing FIFOs
```

```
    if (mkfifo(FIFO1, 0666) < 0 || mkfifo(FIFO2, 0666) < 0) {   // Create
FIFOs
        perror("Error: Failed to create FIFOs");
        exit(EXIT_FAILURE);
    }
```

- Send an array of random numbers to the first FIFO and the second FIFO. Send a command to the second FIFO(requesting a multiplication operation).

```
    srand(time(NULL)); // Seed the random number generator
    int n = atoi(argv[1]);  // Number of elements
    int numbers[n];
    for (int i = 0; i < n; ++i) {
        numbers[i] = rand() % 10; // Generate random numbers between 0 and
9
    }
```

```
    // Parent writes numbers and command to FIFO2
    int fd2 = open(FIFO2, O_WRONLY);
    write(fd2, numbers, n * sizeof(int)); // First send numbers
    write(fd2, "multiply", strlen("multiply") + 1); // Then send command
    close(fd2);
```

- Use the fork() system call to create two child processes and assign each to a FIFO.

```
    pid_t pid1 = fork();
```

Set a signal handler for SIGCHLD in the parent process to handle child process termination.

Implement a zombie protection method

```
    struct sigaction sa; // Signal handler for SIGCHLD
    memset(&sa, 0, sizeof(sa)); // what memset does is that it fills the
first sizeof(sa) bytes of the memory area pointed to by &sa with the
constant byte 0 for the signal handler
    sa.sa_handler = handle_sigchld;
```

```
    sigaction(SIGCHLD, &sa, NULL);  // what SIGCHLD does is that it is
sent to the parent process when a child process terminates
```

The signal handler should call waitpid() to reap the terminated child process, print out the process ID of the exited child, and increment a counter.

```
void handle_sigchld(int sig) {
    int status;
    pid_t pid;
    while ((pid = waitpid(-1, &status, WNOHANG)) > 0) { // WNOHANG: return
immediately if no child has exited
        printf("Debug: Child %d terminated with exit status %d\n", pid,
WEXITSTATUS(status));   // WEXITSTATUS: returns the exit status of the
child
        child_exit_counter++;
    }
}
```

Enter a loop, printing a message containing "proceeding" every two seconds.

```
    while (child_exit_counter < 2) {    // Wait for both children to exit
        printf("Parent: Proceeding, waiting for children to exit...\n");
        sleep(2);
    }
```

OUTPUT EXAMPLE :

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/sistem/hw2$ make
gcc -Wall -o hw2 hw2.c
./hw2 12
Child 1: Read 5, Current Sum: 5
Child 1: Read 4, Current Sum: 9
Child 1: Read 3, Current Sum: 12
Child 1: Read 1, Current Sum: 13
Child 1: Read 7, Current Sum: 20
Child 1: Read 2, Current Sum: 22
Child 1: Read 7, Current Sum: 29
Child 1: Read 3, Current Sum: 32
Child 1: Read 1, Current Sum: 33
Parent: Proceeding, waiting for children to exit...
Child 1: Read 7, Current Sum: 40
Child 1: Read 7, Current Sum: 47
Child 1: Read 1, Current Sum: 48
Child 1: Sum : 48
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Child 2: Multiplication result: 864360
Final Output : multiply + sum : 864408
Debug: Child 355 terminated with exit status 0
Debug: Child 356 terminated with exit status 0
rm -f hw2
```

MEMORY LEAK DOES NOT EXİST

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/sistem/hw2$ valgrind --leak-check=full --show-leak-kinds=all ./hw2 2
==317== Memcheck, a memory error detector
==317== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==317== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==317== Command: ./hw2 2
==317==
Parent: Proceeding, waiting for children to exit...
Child 1: Read 2, Current Sum: 2
Child 1: Read 4, Current Sum: 6
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Parent: Proceeding, waiting for children to exit...
Child 2: Product after multiplication: 8
Child 1: Wrote sum 6 to FIFO2.
Final Output : multiply + sum : 14
==318==
==318== HEAP SUMMARY:
==318==     in use at exit: 0 bytes in 0 blocks
==318==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==318==
==318== All heap blocks were freed -- no leaks are possible
==318==
==318== For lists of detected and suppressed errors, rerun with: -s
==318== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==319==
==319== HEAP SUMMARY:
==319==     in use at exit: 0 bytes in 0 blocks
==319==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==319==
==319== All heap blocks were freed -- no leaks are possible
==319==
==319== For lists of detected and suppressed errors, rerun with: -s
==319== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Debug: Child 318 terminated with exit status 0
Debug: Child 319 terminated with exit status 0
==317==
==317== HEAP SUMMARY:
==317==     in use at exit: 0 bytes in 0 blocks
==317==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==317==
==317== All heap blocks were freed -- no leaks are possible
==317==
==317== For lists of detected and suppressed errors, rerun with: -s
==317== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```