Question 1

Write a program in C which continuously takes integer inputs from the user with scanf function. By using signal() system call for SIGINT, alter the behaviour of Ctrl-C such that whenever the combination is pressed, the program responds with the number of inputs and their average value on screen. Use -1 to exit.

Sample Run

```
$ ./question1
Enter a number: 3
Enter a number: 7
<CTRL-C>
Inputs: 2, Average: 5
Enter a number: 4
Enter a number: 12
Enter a number: 14
<CTRL-C>
Inputs: 5, Average: 8
...
```

Question 2

Execute the following command to create a "data.txt" file which contains 100.000 random letters and numbers:

```
cat /dev/urandom | tr -dc 'a-z0-9' | fold
-w 100000 | head -n 1 > data.txt
```

Now write a C program which creates two child processes. The first child process must count the letters, while the other child process must count the numbers. Both child processes must print their results on screen. In the meanwhile, parent process must display child process ID's on screen and wait for both processes to end.

Sample Run

```
$ ./question2
[PARENT] Child process ID: 3587
[CHILD1] Number of letters: 58392
[PARENT] Child process ID: 3588
[CHILD2] Number of numbers: 41608
```

Question 1

Write a program in C which first initializes an integer variable to a random value between 100 and 200. By using signal() system call for SIGINT, alter the behaviour of Ctrl-C such that whenever this combination is pressed, the program switches between adding or subtracting 10 from this variable in an endless loop with 1 second delays. Your program must indicate whether it is increasing or decreasing the variable and it must end if value of the variable is below 100 or above 200.

Sample Run

```
$ ./question1
[Increasing]
Variable: 188
Variable: 198
<CTRL-C>
[Decreasing]
Variable: 188
Variable: 178
Variable: 168
<CTRL-C>
[Increasing]
Variable: 178
Variable: 188
Variable: 198
Variable: 208
```

Question 2

Write a C program which creates a child process. This child process must write 1.000.000 random integers to a text file, each seperated by one space character and then terminate. Parent process must wait for this child process to terminate, then create an another child process. This new child process must count how many of the numbers in the text file are even and how many of them are odd and then terminate as well. Parent should display process ID's of both child processes and terminate only after both of them are terminated.

Sample Run

```
$ ./question2
[PARENT] Child process ID: 22678
[CHILD1] Wrote 1000000 integers to
numbers.txt, terminating.
[PARENT] Child process ID: 22679
[CHILD2] Even numbers: 682722, odd
numbers: 317278, terminating.
[PARENT] Terminating.
```

Question 1

Write a program in C in which you modify the behaviour of CTRL-C combination using signal() system call for SIGINT, such that the program alternates between generating random letters and random numbers on whenever CTRL-C is pressed. These data must be printed on screen in a loop with 1 second delays. The program must terminate after 15 items are printed on screen.

Sample Run

```
$ ./question1
8
3
4
<CTRL-C>
b
d
j
n
<CTRL-C>
2
7
. . .
```

Question 2

Write a C program which creates two child processes. One of the child processes must open data1.txt and fill it with 10 random integers (0-9). The other child process must open data2.txt and fill it with 10 random letters (a-z). In the meanwhile, parent must print both child processes' id's and wait for them to finish. Then, it must open both data files and read one data from each file alternatively and print it on screen.

Sample Run

```
$ ./question2
[PARENT] Child process ID: 5378
[PARENT] Child process ID: 5379
[CHILD1] Wrote: 6 2 8 6 7 3 3 1 8 4 to
file.
[CHILD2] Wrote: f y n h j w x k p m to
file.
[PARENT] 6 f 2 y 8 n 6 h 7 j 3 w 3 x 1 k
8 p 4 m
```