1. Create a C program that runs a bash command received as a command line argument.

2. Create a C program that generates N random integers (N given at the command line). It then creates a child, sends the numbers via pipe. The child calculates the average and sends the result back.

3. Write a C program that creates two child processes. The two child processes will alternate sending random integers between 1 and 10(inclusively) to one another until one of them sends the number 10. Print messages as the numbers are sent.

4. Write a C program that implements the boltz game. N processes (numbered 1 to N, where N is given) take turns incrementing a number and sending it to the next process. Process 1 starts the game by incrementing the number and sends it to process 2, which increments and sends it to process 3 and so on. Process N will send the number back to process 1. Each process must print the number it sends, unless the number contains the digit 7 or is divisible by 7, in which case it must print "boltz". Implement so that each process has a 1 in 3 chance to fail printing "boltz" when it should, in which case the game stops.

5. Write two C programs that communicate via fifo. Program A is responsible for creating/deleting the fifo. Program A reads commands from the standard input, executes them and sends the output to program B. Program B keeps reading from the fifo and displays whatever it receives at the standard output. This continues until program A receives the "stop" command.

6. Create two processes A and B. A generates a random number n between 50 and 200. If it is even, it sends it to B, if it is odd it sends n+1 to B. B receives the number and divides it by 2 and sends it back to A. The process repeats until n is smaller than 5. The processes will print the value of n at each step.

7. Create two processes A and B. A creates a shared memory segment. A then keeps reading strings from the standard input and places whatever it reads in the shared memory segment(replacing previous data). Process B, on each run, reads the data from the shared memory segment and counts the number of vowels. Process A, upon receiving a SIGINT, deletes the shared memory segment.

8a. Write a C program that reads a matrix from a file. It then creates as many threads as there are rows in the matrix, each thread calculates the sum of the numbers on a row. The main process waits for the threads to finish, then prints the sums.

8b. Same as 9a, but calculate the sum of all the elements of the matrix using as many threads as there are rows, each thread adds to the total the numbers on a row. Use the test matrix to check if the program is calculating the total sum correctly. The expected result is 1000000. Try with and without mutex.

9. Using PIPE channels create and implement the following scenario: Process A reads N integer numbers from the keyboard and sends them another process named B. Process B will add a random number, between 2 and 5, to each received number from process A and will send them to another process named C. The process C will add all the received numbers and will send the result back to process A. All processes will print a debug message before sending and after receiving a number.

10. Write a program that receives strings of alphanumeric characters as command line arguments (validation is not required). For each string the program creates a thread which calculates the number of digits, the number of consonants and the number of vowels. The thread adds the results to three global variables, one for digits, one for vowels and one for consonants. The main program prints the end results (total number of digits, vowels and consonants across all the received command line arguments) and terminates. Use efficient synchronization.

11. Write a C program (we'll refer to it as A) that creates a child process B. Process B generates one random number N between 100 and 1000. Process A keeps generating and sending random numbers between 50 and 1050 to B until the absolute difference between the number generated by A and the number generated by B is less than 50. B prints the generated numbers and all the received numbers. A will print at the end the number of numbers generated until the stop condition was met.

Example:

```
Process B has generated 433

B received 244; difference: 189

B received 367; difference: 66

B received 723; difference: 290

B received 465; difference: 32

Process A has generated 4 numbers
```

12. Write a C program that receives integers as command line argument. The program will keep a frequency vector for all digits. The program will create a thread for each argument that counts the number of occurences of each digit and adds the result to the global frequency vector. Use efficient synchronization.