CSE 3100 Systems Programming	Fall 2025
Lab #5	

Welcome to Lab 5! Read each question carefully before beginning work. Submit .c file to gradescope. A Makefile is provided.

Part 1. (100 points) run2 should use the `fork` and `exec` functions to run two commands specified on the command line. The arguments passed to `run2` are stored in the array `argv`. The first executable is `argv[1]` and it takes one, and only one, argument, which is in `argv[2]`. The second executable is `argv[3]`, and the rest of arguments, if any, are arguments to `argv[3]`.

In the following example, the two commands are `cat Makefile` and `ls a b`.

```
$ ./run2 cat Makefile ls a b
```

There are multiple functions in the `exec` family of functions, which behave in slightly different ways. You should use `execlp` to run the first command, and `execvp` to run the second command. You should read the manpage for `exec`, and understand why these variants are sensible choices here.

If any functions like `fork()` fail, report the error using `perror()` and exit from the process. The parameter to `perror()` is the function name followed by parentheses, as in the following examples.

```
perror("fork()");
perror("execlp()");
```

For each child process successfully created, the parent process waits for the child process to exit and prints out the exit status of the child process using the following format string:

```
"exited=%d exitstatus=%d\n"
```

The first number is from WIFEXITED, and the second is from WEXITSTATUS. Study the demo code or the manuals to learn WIFEXITED and WEXITSTATUS.

Below are some examples of using `run2`. You may have more creative ways to use the program.

```
$ ./run2 vi run2.c make
$ ./run2 touch run2.c make run2
$ ./run2 echo 1 ./run2 echo 2 ./run2 echo 3 ls *
```