Lab Task 4: fork, wait, exit and exec

1. Write a C program that creates a child process using fork(). The parent and child should print their respective process IDs using getpid(). Use getppid() in the child process to print its parent's process ID.

2. Modify the previous program so that the parent waits for the child to finish before exiting. The child should print "Child process running..." and sleep for 2 seconds before exiting.
The parent should wait for the child using wait(), then print "Child process finished, parent exiting."

3.Write a C program that creates two child processes from the parent. Each child process prints its own PID and the parent's PID. The parent waits for both children before exiting.

4.Write a program where the child process prints a message and exits with status 42 using exit(42). The parent process waits for the child and prints the exit status.

Hint: Use WEXITSTATUS(status) after wait(&status).

5. Write a C program that accepts two integers as command-line arguments. Forks a child process. The child adds the two numbers and prints the result.

The parent waits for the child and then prints "Parent process finished."

Example Usage:

```
./program 5  +  7
```

Second argument would be the operator.

6. Modify Exercise 5 so that the child computes the sum and exits using exit(sum). The parent retrieves the child's exit status and prints the sum.

Hint: Use WEXITSTATUS(status).

Expected Output:

```
Parent: Retrieved sum from child: 12
```

7. Write a C program that creates three child processes.
Each child prints its own PID and parent's PID.
The parent waits for all children to finish.

Expected Output:

```
Parent: PID = 1000
Child 1: PID = 1001, Parent PID = 1000
Child 2: PID = 1002, Parent PID = 1000
Child 3: PID = 1003, Parent PID = 1000
Parent: All child processes finished.
```

8.Modify Exercise 7 so that:

The parent does not wait for the children (children become orphaned if the parent exits first).
Each child prints a message and sleeps for 5 seconds before terminating. The parent exits immediately after forking all children.

Expected Output:

```
Parent: Forking children and exiting.
Child 1: Running, PID = 2001, Parent PID = 2000
Child 2: Running, PID = 2002, Parent PID = 2000
Child 3: Running, PID = 2003, Parent PID = 2000
(5 seconds later)
Child 1: Exiting
Child 2: Exiting
Child 3: Exiting
```

9.Write a program that:

Creates a child process. The child process creates its own child (grandchild). The parent waits only for its direct child, while the grandchild runs independently.

Expected Output:

```
Parent: PID = 3000
Child: PID = 3001, Parent PID = 3000
Grandchild: PID = 3002, Parent PID = 3001
Parent: Child 3001 finished.
```

10. Write a program where the parent creates a child.
The child creates another child, and so on, forming a chain of 4 processes. Each process prints its PID and its parent’s PID.

Expected Output:

```
Process 1: PID = 4000, Parent PID = (none)
Process 2: PID = 4001, Parent PID = 4000
Process 3: PID = 4002, Parent PID = 4001
Process 4: PID = 4003, Parent PID = 4002
```

11. Write a C program that creates a large integer array of size 100000000 filled with random numbers and uses a single process to sum all elements. Measures the execution time.

12. Write a C program that Creates a large integer array of size 100000000, filled with random numbers. Spawns four child processes, each computing the sum of a quarter of the array and print it. Parent waits for all the children to finish. Measures the execution time and compare it with exercise 11.

13. Write a program where the child process (after fork()) replaces itself with the ls -la command using execl().

14- Write a program where the child process (after fork()) replaces itself with the lsd command using execl() and show that exec failed (command not found).

15. write a C program (child.c) that prints "Hello from child process!”. Then, write a parent program (parent.c) that Uses fork, The child process replaces itself using exec to run child.c (child.o)

16. Use exec to run the cdf.sh script with arguments (created in lab 2) in a child process.

17. Set an environment variable NAME=“Your Name” and print this variable using echo command using execle.