1) Implement a simple Calculator using system ca#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

void performOperation(char operation, int a, int b) {

int result = 0;

switch (operation) {

case '+':

result = a + b;

printf("Result: %d\n", result);

exit(result);

case '-':

result = a - b;

printf("Result: %d\n", result);

exit(result);

case '\*':

result = a \* b;

printf("Result: %d\n", result);

exit(result);

case '/':

if (b != 0) {

result = a / b;

printf("Result: %d\n", result);

exit(result);

} else {

printf("Error: Division by zero\n");

exit(EXIT\_FAILURE);

}

default:

printf("Invalid operation\n");

exit(EXIT\_FAILURE);

}

}

int main() {

char operation;

int operand1, operand2, status;

printf("Enter an operation (+, -, \*, /): ");

scanf(" %c", &operation); // Notice the space before %c to catch any leading whitespace

printf("Enter the first number: ");

scanf("%d", &operand1);

printf("Enter the second number: ");

scanf("%d", &operand2);

pid\_t pid = fork();

if (pid < 0) {

perror("fork");

return EXIT\_FAILURE;

} else if (pid == 0) {

// Child process

performOperation(operation, operand1, operand2);

} else {

// Parent process waits for child to complete

waitpid(pid, &status, 0);

if (WIFEXITED(status)) {

printf("Child process exited with result = %d\n", WEXITSTATUS(status));

} else {

printf("Child process did not terminate successfully\n");

}

}

return EXIT\_SUCCESS;

}

lls for multi-processing.