#include <stdio.h>

int main() {

int prev\_reading, curr\_reading, total\_units;

float cost\_slab1 = 0, cost\_slab2 = 0, cost\_slab3 = 0, total\_cost = 0, surcharge = 0;

const float maintenance\_charge = 50.0;

printf("Enter the previous meter reading: ");

scanf("%d", &prev\_reading);

printf("Enter the current meter reading: ");

scanf("%d", &curr\_reading);

total\_units = curr\_reading - prev\_reading;

if (total\_units <= 100) {

cost\_slab1 = total\_units \* 3.0;

} else if (total\_units <= 300) {

cost\_slab1 = 100 \* 3.0;

cost\_slab2 = (total\_units - 100) \* 5.0;

} else {

cost\_slab1 = 100 \* 3.0;

cost\_slab2 = 200 \* 5.0;

cost\_slab3 = (total\_units - 300) \* 7.0;

}

total\_cost = cost\_slab1 + cost\_slab2 + cost\_slab3 + maintenance\_charge;

if (total\_cost > 1000) {

surcharge = total\_cost \* 0.10;

}

total\_cost += surcharge;

printf("\nElectricity Bill Summary\n");

printf("---------------------------------------------\n");

printf("Slab | Rate | Consumed | Cost\n");

printf("---------------------------------------------\n");

printf("1 (<100) | @3.00 | %8d | %8.2f\n", (total\_units > 100 ? 100 : total\_units), cost\_slab1);

printf("2 (101-300) | @5.00 | %8d | %8.2f\n", (total\_units > 300 ? 200 : (total\_units > 100 ? total\_units - 100 : 0)), cost\_slab2);

printf("3 (>300) | @7.00 | %8d | %8.2f\n", (total\_units > 300 ? total\_units - 300 : 0), cost\_slab3);

printf("Maintenance | Fixed | 1 | %8.2f\n", maintenance\_charge);

if (surcharge > 0) {

printf("Surcharge @10%% | | | %8.2f\n", surcharge);

}

printf("---------------------------------------------\n");

printf("Total Bill | | | %8.2f\n", total\_cost);

printf("---------------------------------------------\n");

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#define FILE\_NAME "input.txt"

int count\_words(const char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

perror("Unable to open file");

exit(EXIT\_FAILURE);

}

int count = 0;

char word[256];

while (fscanf(file, "%s", word) == 1) {

count++;

}

fclose(file);

return count;

}

int count\_lines(const char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

perror("Unable to open file");

exit(EXIT\_FAILURE);

}

int count = 0;

char ch;

while ((ch = fgetc(file)) != EOF) {

if (ch == '\n') {

count++;

}

}

fclose(file);

return count;

}

int count\_characters(const char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

perror("Unable to open file");

exit(EXIT\_FAILURE);

}

int count = 0;

while (fgetc(file) != EOF) {

count++;

}

fclose(file);

return count;

}

int main() {

pid\_t p1, p2, p3;

p1 = fork();

if (p1 == 0) {

int word\_count = count\_words(FILE\_NAME);

printf("Child 1 (Words): %d\n", word\_count);

exit(EXIT\_SUCCESS);

}

p2 = fork();

if (p2 == 0) {

int line\_count = count\_lines(FILE\_NAME);

printf("Child 2 (Lines): %d\n", line\_count);

exit(EXIT\_SUCCESS);

}

p3 = fork();

if (p3 == 0) {

int char\_count = count\_characters(FILE\_NAME);

printf("Child 3 (Characters): %d\n", char\_count);

exit(EXIT\_SUCCESS);

}

wait(NULL);

wait(NULL);

wait(NULL);

printf("Parent: All child processes have finished.\n");

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <semaphore.h>

#include <fcntl.h>

#define TERM\_COUNT 10

void calculate\_and\_print\_series(int id, sem\_t\* sem\_current, sem\_t\* sem\_next) {

for (int k = 0; k < TERM\_COUNT; k++) {

sem\_wait(sem\_current);

int term = 3 \* k + id;

printf("%d ", term);

sem\_post(sem\_next);

}

}

int main() {

pid\_t p1, p2, p3;

setvbuf(stdout, NULL, \_IONBF, 0);

sem\_t\* sem1 = sem\_open("/sem1", O\_CREAT | O\_EXCL, 0666, 1);

sem\_t\* sem2 = sem\_open("/sem2", O\_CREAT | O\_EXCL, 0666, 0);

sem\_t\* sem3 = sem\_open("/sem3", O\_CREAT | O\_EXCL, 0666, 0);

if (sem1 == SEM\_FAILED || sem2 == SEM\_FAILED || sem3 == SEM\_FAILED) {

perror("sem\_open failed");

exit(EXIT\_FAILURE);

}

p1 = fork();

if (p1 == 0) {

calculate\_and\_print\_series(1, sem1, sem2);

exit(EXIT\_SUCCESS);

}

p2 = fork();

if (p2 == 0) {

calculate\_and\_print\_series(2, sem2, sem3);

exit(EXIT\_SUCCESS);

}

p3 = fork();

if (p3 == 0) {

calculate\_and\_print\_series(3, sem3, sem1);

exit(EXIT\_SUCCESS);

}

wait(NULL);

wait(NULL);

wait(NULL);

printf("\n");

sem\_close(sem1);

sem\_close(sem2);

sem\_close(sem3);

sem\_unlink("/sem1");

sem\_unlink("/sem2");

sem\_unlink("/sem3");

return 0;

}

gcc Q3.c -lpthread