[Q1: Process Synchronization using Mutex Locks](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper" \l "q1-process-synchronization-using-mutex-locks-1)

#include <stdio.h>

#include <pthread.h>

pthread\_mutex\_t mutex;

int shared\_variable = 0;

void \*increment(void \*arg) {

for (int i = 0; i < 5; ++i) {

pthread\_mutex\_lock(&mutex);

shared\_variable++;

printf("Thread %s: %d\n", (char \*)arg, shared\_variable);

pthread\_mutex\_unlock(&mutex);

}

return NULL;

}

int main() {

pthread\_t thread1, thread2;

pthread\_mutex\_init(&mutex, NULL);

pthread\_create(&thread1, NULL, increment, "A");

pthread\_create(&thread2, NULL, increment, "B");

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

pthread\_mutex\_destroy(&mutex);

return 0;

}

[Q2: Open File in Read-Only Mode and Read the Last 5 Characters](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper#q2-open-file-in-read-only-mode-and-read-the-last-5-characters-1)

#include <stdio.h>

int main() {

FILE \*file = fopen("example.txt", "r");

if (file != NULL) {

fseek(file, -5, SEEK\_END);

char buffer[6]; // 5 characters + null terminator

fread(buffer, sizeof(char), 5, file);

buffer[5] = '\0';

printf("Last 5 characters: %s\n", buffer);

fclose(file);

} else {

printf("Error opening the file.\n");

}

return 0;

}

[Q3: Create a File and Write "Hello" after 4 Characters](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper#q3-create-a-file-and-write-hello-after-4-characters-1)

#include <fcntl.h>

#include <unistd.h>

int main() {

int fd = open("example.txt", O\_WRONLY | O\_CREAT, 0644);

if (fd != -1) {

lseek(fd, 4, SEEK\_SET); // Move the cursor to the 4th position

write(fd, "hello", 5); // Write "hello"

close(fd);

}

return 0;

}

[Q4: Calculate Addition in Parent Process, Display Result in Child Process](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper#q4-calculate-addition-in-parent-process-display-result-in-child-process-1)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

int fd[2];

pipe(fd);

pid\_t pid = fork();

if (pid == 0) { // Child process

close(fd[1]); // Close write end

int result;

read(fd[0], &result, sizeof(result));

close(fd[0]); // Close read end

printf("Child process: Sum is %d\n", result);

} else if (pid > 0) { // Parent process

close(fd[0]); // Close read end

int num1 = 10, num2 = 20, sum;

sum = num1 + num2;

write(fd[1], &sum, sizeof(sum));

close(fd[1]); // Close write end

}

return 0;

}

[Q5: Write into a Pipe using popen() and pclose()](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper#q5-write-into-a-pipe-using-popen-and-pclose)

#include <stdio.h>

int main() {

FILE \*pipe\_fp;

char buffer[20];

pipe\_fp = popen("echo 'Hello, Pipe!'", "r");

if (pipe\_fp != NULL) {

fread(buffer, sizeof(char), sizeof(buffer), pipe\_fp);

printf("Received from pipe: %s\n", buffer);

pclose(pipe\_fp);

} else {

printf("Error opening pipe.\n");

}

return 0;

}

[Q6: Create Two Threads - Print Numbers and Check Even/Odd](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper#q6-create-two-threads---print-numbers-and-check-evenodd-1)

#include <stdio.h>

#include <pthread.h>

void \*printNumbers(void \*arg) {

for (int i = 1; i <= 10; ++i) {

printf("Thread 1: %d\n", i);

}

return NULL;

}

void \*checkEvenOdd(void \*arg) {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (num % 2 == 0) {

printf("Thread 2: Even\n");

} else {

printf("Thread 2: Odd\n");

}

return NULL;

}

int main() {

pthread\_t thread1, thread2;

pthread\_create(&thread1, NULL, printNumbers, NULL);

pthread\_create(&thread2, NULL, checkEvenOdd, NULL);

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

return 0;

}

[Q7: Read from 3rd to 10th Character using System Calls](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper#q7-read-from-3rd-to-10th-character-using-system-calls-1)

#include <fcntl.h>

#include <unistd.h>

int main() {

int fd = open("example.txt", O\_RDONLY);

if (fd != -1) {

lseek(fd, 2, SEEK\_SET); // Move the cursor to the 3rd character

char buffer[9]; // 8 characters + null terminator

read(fd, buffer, sizeof(buffer) - 1);

buffer[8] = '\0';

printf("Characters 3 to 10: %s\n", buffer);

close(fd);

} else {

printf("Error opening the file.\n");

}

return 0;

}[Q8: Process Hierarchy - P1 has children P2 and P3](https://github.com/AnshulChoudhary001/CSE-325-Question-Paper" \l "q8-process-hierarchy---p1-has-children-p2-and-p3-1)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

pid\_t pid1, pid2, pid3;

pid1 = getpid();

if ((pid2 = fork()) == 0) {

// Child process P2

printf("P2: PID=%d, Parent PID=%d\n", getpid(), getppid());

} else if (pid2 > 0) {

// Parent process P1

if ((pid3 = fork()) == 0) {

// Child process P3

printf("P3: PID=%d, Parent PID=%d\n", getpid(), getppid());

} else if (pid3 > 0) {

// Parent process P1

printf("P1: PID=%d\n", pid1);

sleep(2); // Ensure P2 and P3 print before P1

} else {

perror("Error creating P3");

exit(EXIT\_FAILURE);

}

} else {

perror("Error creating P2");

exit(EXIT\_FAILURE);

}

return 0;

}.

Q9:Hierarchy with p1->p2->p3

#include<iostream>#include<stdlib.h>#include<unistd.h> int main(){pid\_t pid1,pid2,pid3; pid1=getid(); if((pid2= fork()) ==0){printf(“P2: PID=%d,Parent pid =%d”)