# TASK 1:

# #include <stdio.h>

# #include <stdlib.h>

# #include <pthread.h>

# #include <unistd.h>

# pthread\_mutex\_t lock;

# int num\_cars;

# int num\_boats;

# int time\_per\_car;

# int time\_per\_boat;

# int bridge\_open = 0;

# void\* car\_thread(void\* arg)

# {

# int car\_id = \*(int\*)arg;

# printf("Car %d is waiting to cross the bridge\n", car\_id);

# pthread\_mutex\_lock(&lock);

# while (bridge\_open)

# {

# printf("Car %d is waiting for the bridge to close\n", car\_id);

# pthread\_mutex\_unlock(&lock);

# sleep(1);

# pthread\_mutex\_lock(&lock);

# }

# printf("Car %d is crossing the bridge\n", car\_id);

# bridge\_open = 1;

# pthread\_mutex\_unlock(&lock);

# sleep(time\_per\_car);

# bridge\_open = 0;

# printf("Car %d has crossed the bridge\n", car\_id);

# pthread\_exit(NULL);

# }

# void\* boat\_thread(void\* arg)

# {

# int boat\_id = \*(int\*)arg;

# printf("Boat %d is waiting to cross the bridge\n", boat\_id);

# pthread\_mutex\_lock(&lock);

# while (bridge\_open)

# {

# printf("Boat %d is waiting for the bridge to close\n", boat\_id);

# pthread\_mutex\_unlock(&lock);

# sleep(1);

# pthread\_mutex\_lock(&lock);

# }

# printf("Boat %d is crossing the bridge\n", boat\_id);

# bridge\_open = 1;

# pthread\_mutex\_unlock(&lock);

# sleep(time\_per\_boat);

# bridge\_open = 0;

# printf("Boat %d has crossed the bridge\n", boat\_id);

# pthread\_exit(NULL);

# }

# int main()

# {

# printf("Enter number of cars: ");

# scanf("%d", &num\_cars);

# printf("Enter number of boats: ");

# scanf("%d", &num\_boats);

# printf("Enter time per car (in seconds): ");

# scanf("%d", &time\_per\_car);

# printf("Enter time per boat (in seconds): ");

# scanf("%d", &time\_per\_boat);

# pthread\_t car\_threads[num\_cars];

# pthread\_t boat\_threads[num\_boats];

# int car\_ids[num\_cars];

# int boat\_ids[num\_boats];

# pthread\_mutex\_init(&lock, NULL);

# 

# for (int i = 0; i < num\_cars; i++)

# {

# car\_ids[i] = i+1;

# pthread\_create(&car\_threads[i], NULL, car\_thread, &car\_ids[i]);

# }

# for (int i = 0; i < num\_boats; i++)

# {

# boat\_ids[i] = i+1;

# pthread\_create(&boat\_threads[i], NULL, boat\_thread, &boat\_ids[i]);

# }

# for (int i = 0; i < num\_cars; i++)

# {

# pthread\_join(car\_threads[i], NULL);

# }

# for (int i = 0; i < num\_boats; i++)

# {

# pthread\_join(boat\_threads[i], NULL);

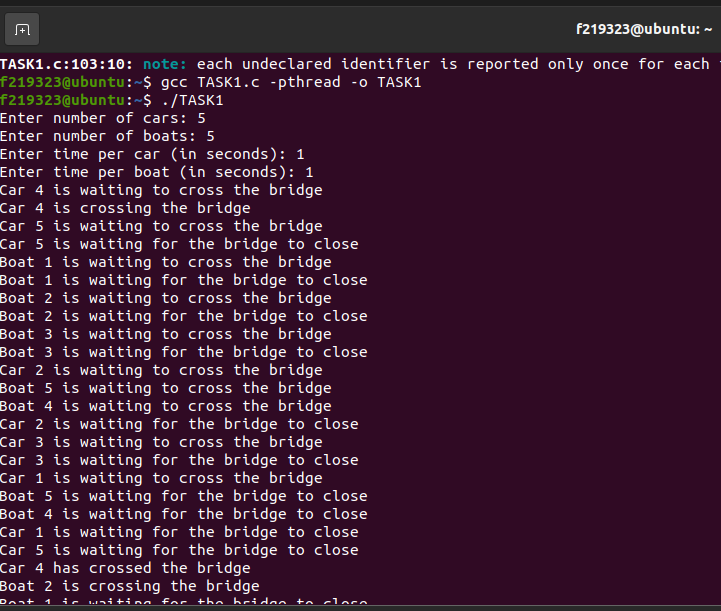
# }

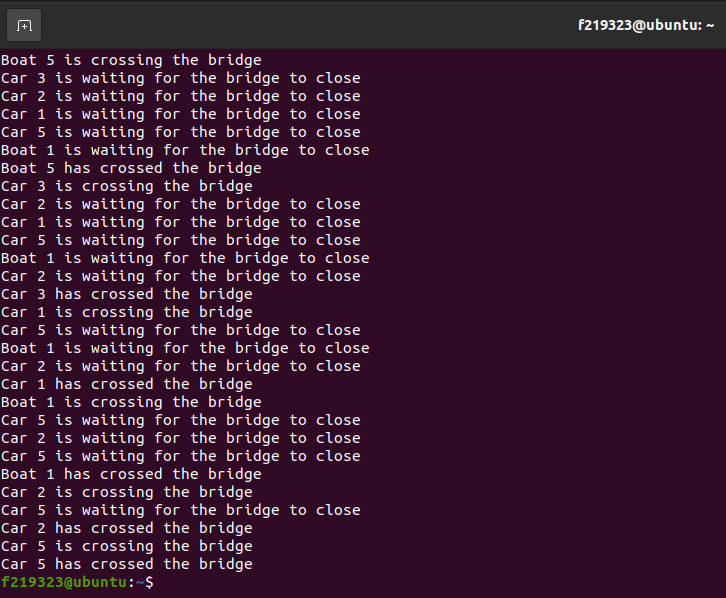
# pthread\_mutex\_destroy(&lock);

# return 0;

# }

# OUTPUT:





TASK 2:  
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <semaphore.h>

sem\_t bridge\_sem;

int num\_cars;

int num\_boats;

int time\_per\_car;

int time\_per\_boat;

int bridge\_open = 0;

void\* car\_thread(void\* arg)

{

int car\_id = \*(int\*)arg;

printf("Car %d is waiting to cross the bridge\n", car\_id);

while (1)

{

sem\_wait(&bridge\_sem);

if (!bridge\_open)

{

printf("Car %d is crossing the bridge\n", car\_id);

bridge\_open = 1;

sem\_post(&bridge\_sem);

sleep(time\_per\_car);

bridge\_open = 0;

printf("Car %d has crossed the bridge\n", car\_id);

break;

}

else

{

sem\_post(&bridge\_sem);

printf("Car %d is waiting for the bridge to close\n", car\_id);

sleep(1);

}

}

pthread\_exit(NULL);

}

void\* boat\_thread(void\* arg)

{

int boat\_id = \*(int\*)arg;

printf("Boat %d is waiting to cross the bridge\n", boat\_id);

while (1)

{

sem\_wait(&bridge\_sem);

if (!bridge\_open)

{

printf("Boat %d is crossing the bridge\n", boat\_id);

bridge\_open = 1;

sem\_post(&bridge\_sem);

sleep(time\_per\_boat);

bridge\_open = 0;

printf("Boat %d has crossed the bridge\n", boat\_id);

break;

}

else

{

sem\_post(&bridge\_sem);

printf("Boat %d is waiting for the bridge to close\n", boat\_id);

sleep(1);

}

}

pthread\_exit(NULL);

}

int main()

{

printf("Enter number of cars: ");

scanf("%d", &num\_cars);

printf("Enter number of boats: ");

scanf("%d", &num\_boats);

printf("Enter time per car (in seconds): ");

scanf("%d", &time\_per\_car);

printf("Enter time per boat (in seconds): ");

scanf("%d", &time\_per\_boat);

pthread\_t car\_threads[num\_cars];

pthread\_t boat\_threads[num\_boats];

int car\_ids[num\_cars];

int boat\_ids[num\_boats];

sem\_init(&bridge\_sem, 0, 1);

for (int i = 0; i < num\_cars; i++)

{

car\_ids[i] = i+1;

pthread\_create(&car\_threads[i], NULL, car\_thread, &car\_ids[i]);

}

for (int i = 0; i < num\_boats; i++)

{

boat\_ids[i] = i+1;

pthread\_create(&boat\_threads[i], NULL, boat\_thread, &boat\_ids[i]);

}

for (int i = 0; i < num\_cars; i++)

{

pthread\_join(car\_threads[i], NULL);

}

for (int i = 0; i < num\_boats; i++)

{

pthread\_join(boat\_threads[i], NULL);

}

sem\_destroy(&bridge\_sem);

return 0;

}



TASK 3:  
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <semaphore.h>

sem\_t bridge\_sem;

sem\_t car\_sem;

sem\_t boat\_sem;

int num\_cars;

int num\_boats;

int time\_per\_car;

int time\_per\_boat;

int bridge\_open = 0;

void\* car\_thread(void\* arg)

{

int car\_id = \*(int\*)arg;

printf("Car %d is waiting to cross the bridge\n", car\_id);

while (1)

{

sem\_wait(&car\_sem);

sem\_wait(&bridge\_sem);

if (!bridge\_open)

{

printf("Car %d is crossing the bridge\n", car\_id);

bridge\_open = 1;

sem\_post(&bridge\_sem);

sem\_post(&car\_sem);

sleep(time\_per\_car);

bridge\_open = 0;

printf("Car %d has crossed the bridge\n", car\_id);

break;

}

else

{

sem\_post(&bridge\_sem);

sem\_post(&car\_sem);

printf("Car %d is waiting for the bridge to close\n", car\_id);

sleep(1);

}

}

pthread\_exit(NULL);

}

void\* boat\_thread(void\* arg)

{

int boat\_id = \*(int\*)arg;

printf("Boat %d is waiting to cross the bridge\n", boat\_id);

while (1)

{

sem\_wait(&boat\_sem);

sem\_wait(&bridge\_sem);

if (!bridge\_open)

{

printf("Boat %d is crossing the bridge\n", boat\_id);

bridge\_open = 1;

sem\_post(&bridge\_sem);

sem\_post(&boat\_sem);

sleep(time\_per\_boat);

bridge\_open = 0;

printf("Boat %d has crossed the bridge\n", boat\_id);

break;

}

else

{

sem\_post(&bridge\_sem);

sem\_post(&boat\_sem);

printf("Boat %d is waiting for the bridge to close\n", boat\_id);

sleep(1);

}

}

pthread\_exit(NULL);

}

int main()

{

printf("Enter number of cars: ");

scanf("%d", &num\_cars);

printf("Enter number of boats: ");

scanf("%d", &num\_boats);

printf("Enter time per car (in seconds): ");

scanf("%d", &time\_per\_car);

printf("Enter time per boat (in seconds): ");

scanf("%d", &time\_per\_boat);

pthread\_t car\_threads[num\_cars];

pthread\_t boat\_threads[num\_boats];

int car\_ids[num\_cars];

int boat\_ids[num\_boats];

sem\_init(&bridge\_sem, 0, 1);

sem\_init(&car\_sem, 0, 2);

sem\_init(&boat\_sem, 0, 3);

for (int i = 0; i < num\_cars; i++)

{

car\_ids[i] = i+1;

pthread\_create(&car\_threads[i], NULL, car\_thread, &car\_ids[i]);

}

for (int i = 0; i < num\_boats; i++)

{

boat\_ids[i] = i+1;

pthread\_create(&boat\_threads[i], NULL, boat\_thread, &boat\_ids[i]);

}

for (int i = 0; i < num\_cars; i++)

{

pthread\_join(car\_threads[i], NULL);

}

for (int i = 0; i < num\_boats; i++)

{

pthread\_join(boat\_threads[i], NULL);

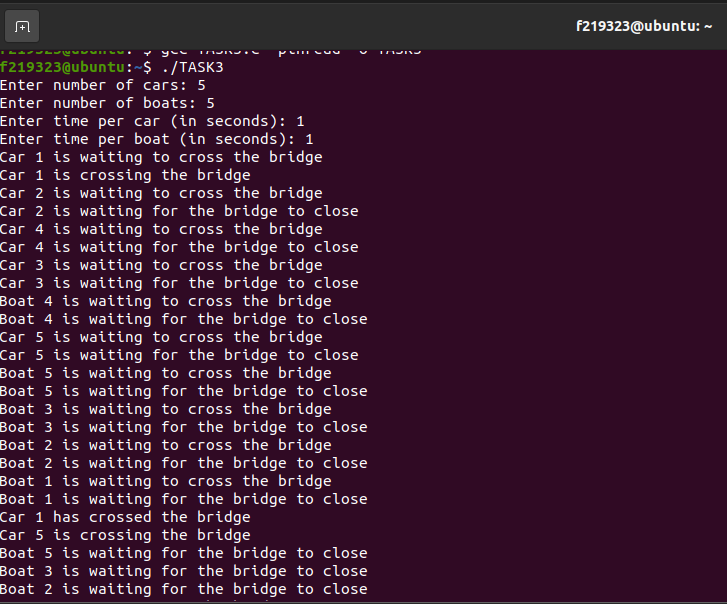
}

sem\_destroy(&bridge\_sem);

return 0;

}

# OUTPUT:



# TASK 4:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

int fuel\_left = 100;

pthread\_mutex\_t mutex\_fuel;

pthread\_cond\_t cond\_fuel\_not\_empty;

void \*fuel\_station()

{

while (1)

{

pthread\_mutex\_lock(&mutex\_fuel);

if (fuel\_left < 100)

{

fuel\_left++;

printf("Fuel Station: Produced 1 liter of fuel, fuel left: %d\n", fuel\_left);

pthread\_cond\_signal(&cond\_fuel\_not\_empty);

}

pthread\_mutex\_unlock(&mutex\_fuel);

usleep(100000);

}

return NULL;

}

void \*car()

{

while (1)

{

pthread\_mutex\_lock(&mutex\_fuel);

while (fuel\_left < 40)

{

printf("Car: Waiting for fuel, fuel left: %d\n", fuel\_left);

pthread\_cond\_wait(&cond\_fuel\_not\_empty, &mutex\_fuel);

}

fuel\_left -= 40;

printf("Car: Consumed %d liters of fuel, fuel left: %d\n", 40, fuel\_left);

pthread\_mutex\_unlock(&mutex\_fuel);

usleep(500000);

}

return NULL;

}

int main()

{

pthread\_t fuel\_station\_thread, car\_thread1, car\_thread2;

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

pthread\_cond\_init(&cond\_fuel\_not\_empty, NULL);

pthread\_create(&fuel\_station\_thread, NULL, fuel\_station, NULL);

pthread\_create(&car\_thread1, NULL, car, NULL);

pthread\_create(&car\_thread2, NULL, car, NULL);

pthread\_join(fuel\_station\_thread, NULL);

pthread\_join(car\_thread1, NULL);

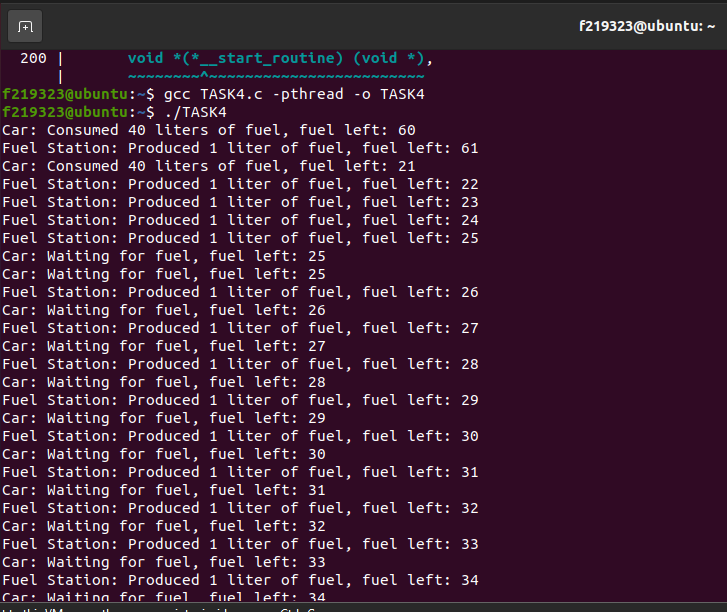
pthread\_join(car\_thread2, NULL);

pthread\_mutex\_destroy(&mutex\_fuel);

pthread\_cond\_destroy(&cond\_fuel\_not\_empty);

return 0;

}

OUTPUT:  


# TASK 5:

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

sem\_t mutex;

void \*job\_i()

{

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

{

sem\_wait(&mutex);

printf("Job i is running\n");

sem\_post(&mutex);

}

return NULL;

}

void \*job\_j()

{

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

{

sem\_wait(&mutex);

printf("Job j is running\n");

sem\_post(&mutex);

}

return NULL;

}

int main()

{

pthread\_t job\_i\_thread, job\_j\_thread;

sem\_init(&mutex, 0, 1);

pthread\_create(&job\_i\_thread, NULL, job\_i, NULL);

pthread\_create(&job\_j\_thread, NULL, job\_j, NULL);

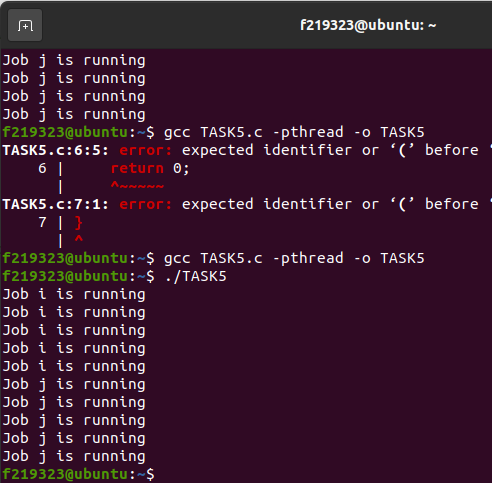
pthread\_join(job\_i\_thread, NULL);

pthread\_join(job\_j\_thread, NULL);

sem\_destroy(&mutex);

return 0;

}

OUTPUT:  


# TASK 6:

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

sem\_t mutex, full, empty;

int buffer[5];

int in = 0, out = 0;

void\* producer()

{

int item;

while (1)

{

printf("Producer is waiting...\n");

sleep(1);

printf("Producer is adding an item to the buffer...\n");

item = rand() % 100;

sem\_wait(&empty);

sem\_wait(&mutex);

buffer[in] = item;

in = (in + 1) % 5;

printf("Producer added %d to the buffer.\n", item);

sem\_post(&mutex);

sem\_post(&full);

}

pthread\_exit(NULL);

}

void\* consumer()

{

int item;

while (1)

{

printf("Consumer is waiting...\n");

sleep(1);

sem\_wait(&full);

sem\_wait(&mutex);

item = buffer[out];

out = (out + 1) % 5;

printf("Consumer removed %d from the buffer.\n", item);

sem\_post(&mutex);

sem\_post(&empty);

printf("Consumer is using the item %d...\n", item);

}

pthread\_exit(NULL);

}

int main()

{

int choice;

pthread\_t tid1, tid2;

sem\_init(&mutex, 0, 1);

sem\_init(&full, 0, 0);

sem\_init(&empty, 0, 5);

printf("Enter 1 for Producer and 2 for Consumer: ");

scanf("%d", &choice);

if (choice == 1)

{

printf("Starting Producer...\n");

pthread\_create(&tid1, NULL, producer, NULL);

pthread\_join(tid1, NULL);

}

else if (choice == 2)

{

printf("Starting Consumer...\n");

pthread\_create(&tid2, NULL, consumer, NULL);

pthread\_join(tid2, NULL);

}

else

{

printf("Invalid choice.\n");

}

sem\_destroy(&mutex);

sem\_destroy(&full);

sem\_destroy(&empty);

return 0;

# }

# OUTPUT:

