**CPP LAB**

**ASSIGNMENT-4**

**NAME-**ABHISHEK TIBREWAL

**ID-**2016UCP1103

**BATCH-**A (1, 2)

1. **Implement solution of Critical Section problem withSemaphores (two processes).**

**CODE:**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

sem\_t s;

void\* thread(void\* arg)

{

char\* msg=(char\*)arg;

while(1)

{

//wait

sem\_wait(&s);

//printf("\n%s",msg);

printf("\n%s:Entered critical section",msg);

//critical section

sleep(2);

//signal

printf("\n%s:Exiting critical section\n",msg);

sem\_post(&s);

}

}

int main()

{

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

pthread\_t t1,t2;

char\* m1="this is thread1";

char\* m2="this is thread2";

pthread\_create(&t1,NULL,thread,(void \*)m1);

sleep(2);

pthread\_create(&t2,NULL,thread,(void \*)m2);

pthread\_join(t1,NULL);

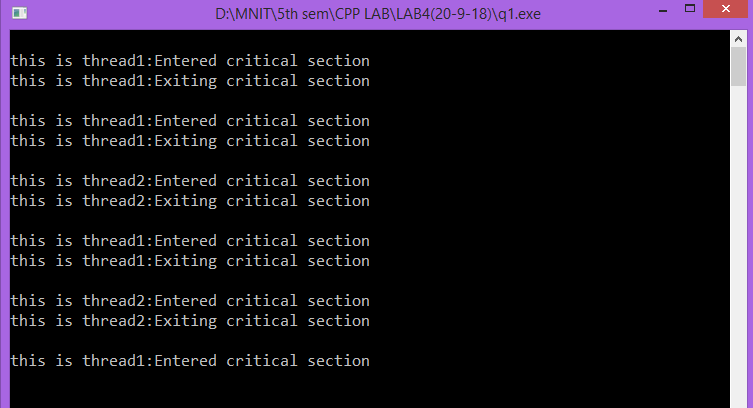
pthread\_join(t2,NULL);

sem\_destroy(&s);

return 0;

}

**OUTPUT:**

****

1. **Implement solution of Critical Section problem with Semaphores (N processes).**

**CODE:**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

sem\_t s;

void\* thread(void\* arg)

{

int x=(int)arg;

while(1)

{

sem\_wait(&s);

printf("\nTHREAD %d:Entered critical section",x);

sleep(2);

printf("\nTHREAD %d:Exiting critical section\n",x);

sem\_post(&s);

}

}

int main()

{

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

pthread\_t t[10];

int i=0;

for(i=0;i<10;i++)

{

pthread\_create(&t[i],NULL,thread,(void \*)i);

sleep(2);

}

for(i=0;i<10;i++)

{

pthread\_join(t[i],NULL);

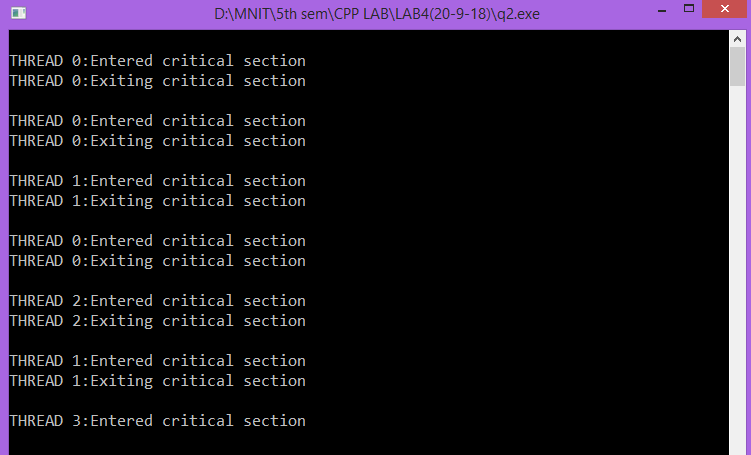
}

sem\_destroy(&s);

return 0;

}

**OUTPUT:**

****

1. **Implement producer-consumer problem with Semaphores (infinite buffer).**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include <pthread.h>

#include <semaphore.h>

#define SIZE 100000

int que[SIZE];

int front,rear;

sem\_t empty, products;

void \*producer(void \*arg)

{

int value;

do

{

int value = rand() % 100;

printf("produced products: %d\n", value);

//sem\_wait(&empty); //wait for empty space in queue

que[rear] = value;

rear = (rear + 1) % SIZE;

sem\_post(&products);

sleep(2);

}

while(value);

pthread\_exit(NULL);

}

void \*consumer(void \*arg)

{

int value;

do

{

sem\_wait(&products); //wait for some product in queue

value = que[front];

front = (front + 1) % SIZE;

printf("consumed products: %d\n", value);

//sem\_post(&empty);

sleep(2);

}

while(value);

pthread\_exit(NULL);

}

int main(int argc, char \*argv[])

{

pthread\_t prod\_t, cons\_t;

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

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

pthread\_create(&prod\_t, NULL, producer, NULL);

pthread\_create(&cons\_t, NULL, consumer, NULL);

pthread\_join(prod\_t, NULL);

pthread\_join(cons\_t, NULL);

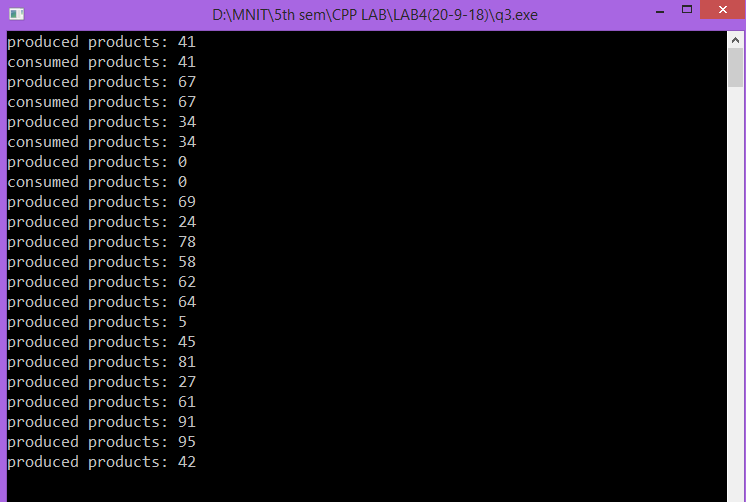
sem\_destroy(&empty);

sem\_destroy(&products);

return 0;

}

**OUTPUT:**

****

1. **Implement producer-consumer problem with Semaphores (finite buffer).**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include <pthread.h>

#include <semaphore.h>

#define SIZE 10

int que[SIZE];

int front,rear;

sem\_t empty, products;

void \*producer(void \*arg)

{

int value;

do

{

int value = rand() % 100;

printf("produced products: %d\n", value);

sem\_wait(&empty); //wait for empty space in queue

que[rear] = value;

rear = (rear + 1) % SIZE;

sem\_post(&products);

sleep(2);

}

while(value);

pthread\_exit(NULL);

}

void \*consumer(void \*arg)

{

int value;

do

{

sem\_wait(&products); //wait for some product in queue

value = que[front];

front = (front + 1) % SIZE;

printf("consumed products: %d\n", value);

sem\_post(&empty);

sleep(2);

}

while(value);

pthread\_exit(NULL);

}

int main(int argc, char \*argv[])

{

pthread\_t prod\_t, cons\_t;

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

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

pthread\_create(&prod\_t, NULL, producer, NULL);

pthread\_create(&cons\_t, NULL, consumer, NULL);

pthread\_join(prod\_t, NULL);

pthread\_join(cons\_t, NULL);

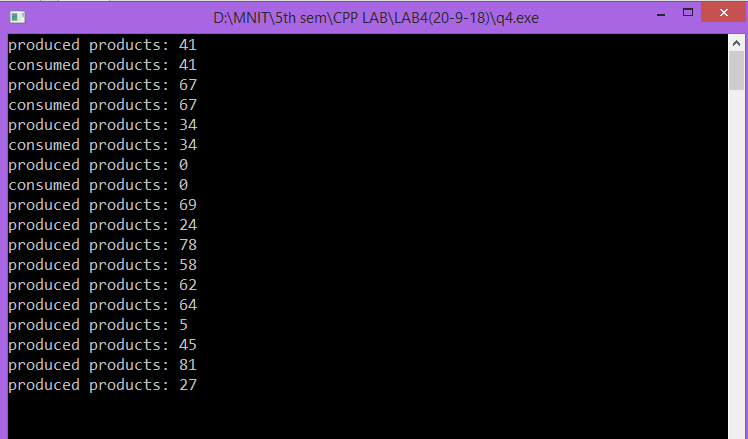
sem\_destroy(&empty);

sem\_destroy(&products);

return 0;

}

**OUTPUT:**

****