SYNCHRONIZATION

NAME	S.SHANTHAN REDDY
ROLL No.	CB.EN.U4CSE19459

```c
#include <windows.h>
#include <stdio.h>

#define MAX_SEM_COUNT 4
#define THREADCOUNT 4

HANDLE ghSemaphore;
int num1, num2, sum;

DWORD WINAPI ThreadProcSemaphore( LPVOID lpParam )
{

    // lpParam not used in this example
    UNREFERENCED_PARAMETER(lpParam);

    DWORD dwWaitResult;
    BOOL bContinue=TRUE;

    while(bContinue)
    {
        // Try to enter the semaphore gate.

        dwWaitResult = WaitForSingleObject(
            ghSemaphore,   // handle to semaphore
            INFINITE);          // zero-second time-out interval

        switch (dwWaitResult)
        {
            // The semaphore object was signaled.
            case WAIT_OBJECT_0:
                // TODO: Perform
            printf("\nThread with %d id is executing...",
GetCurrentThreadId());
            printf("\nEnter 1st number: ");
            scanf("%d", &num1);

            printf("Enter 2nd number: ");
            scanf("%d", &num2);

            sum = num1 + num2;
            printf("Sum of %d and %d : %d\n", num1, num2, sum);


            bContinue=FALSE;

                // Release the semaphore when task is finished
```

```c
            if (!ReleaseSemaphore(
                    ghSemaphore,  // handle to semaphore
                    1,            // increase count by one
                    NULL) )       // not interested in previous count
            {
                printf("ReleaseSemaphore error: %d\n", GetLastError());
            }
            break;

        // The semaphore was nonsignaled, so a time-out occurred.
        case WAIT_TIMEOUT:
            printf("Thread %d: wait timed out\n",
GetCurrentThreadId());
            break;
        }
    }
    return TRUE;
}

DWORD WINAPI ThreadProc( LPVOID lpParam )
{

    // lpParam not used in this example
    UNREFERENCED_PARAMETER(lpParam);
    int i;

    for( i=0; i < 5; i++)
    {
       Sleep(2);
       printf("Thread with %d id is executing..... \n",
GetCurrentThreadId());
    }

    /*printf("\nEnter 1st number: ");
    scanf("%d", &num1);

   printf("\nEnter 2nd number: ");
    scanf("%d", &num2);

    sum = num1 + num2;

    printf("Sum : %d", sum);*/

    return 0;
}

int main( void )
{
    HANDLE aThread[THREADCOUNT];
    DWORD ThreadID;
    int i;

    // Create a semaphore with initial and max counts of MAX_SEM_COUNT

    ghSemaphore = CreateSemaphore(
        NULL,           // default security attributes
        1,              // initial value
        1,              // no.of resources
        NULL);          // unnamed semaphore
```

```c
    if (ghSemaphore == NULL)
    {
        printf("CreateSemaphore error: %d\n", GetLastError());
        return 1;
    }

    // Create worker threads

    printf("\nTHREAD SYNCHRONIZATION WITH SEMAPHORES\n");
    for( i=0; i < 2; i++ )
    {
        aThread[i] = CreateThread(
                        NULL,        // default security attributes
                        0,           // default stack size
                        &ThreadProcSemaphore,        //starting address of the
thread
                        NULL,        // no thread function arguments
                        0,           // default creation flags
                        &ThreadID); // receive thread identifier

        if( aThread[i] == NULL )
        {
            printf("CreateThread error: %d\n", GetLastError());
            return 1;
        }
    }

    // Wait for all threads to terminate

    WaitForMultipleObjects(
        2,          // number of object handles in the array
        aThread,         // array of object handles
        TRUE,            // function returns when the state of all objects in
the handles array is signaled
        INFINITE);      // time-out interval, in milliseconds

    // Close thread and semaphore handles

    for( i=0; i < 2; i++ )
        CloseHandle(aThread[i]);

    CloseHandle(ghSemaphore);


    printf("\nTHREADS WITHOUT SEMAPHORES\n");
    for( i=2; i < 4; i++ )
    {
        aThread[i] = CreateThread(
                        NULL,        // default security attributes
                        0,           // default stack size
                        (LPTHREAD_START_ROUTINE) ThreadProc,     //starting
address of the thread
                        NULL,        // no thread function arguments
                        0,           // default creation flags
                        &ThreadID); // receive thread identifier

        if( aThread[i] == NULL )
        {
            printf("CreateThread error: %d\n", GetLastError());
            return 1;
        }
```

```
    }

  WaitForSingleObject(
    aThread[2],    // handle to semaphore
      INFINITE);           // zero-second time-out interval

  WaitForSingleObject(
      aThread[3],   // handle to semaphore
      INFINITE);           // zero-second time-out interval


   return 0;
}
```

Output :


```
C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.19042.906]
(c) Microsoft Corporation. All rights reserved.

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>gcc semaphore.cpp

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>a

THREAD SYNCHRONIZATION WITH SEMAPHORES

Thread with 17540 id is executing...
Enter 1st number: 59
Enter 2nd number: 53
Sum of 59 and 53 : 112

Thread with 18044 id is executing...
Enter 1st number: 44
Enter 2nd number: 49
Sum of 44 and 49 : 93

THREADS WITHOUT SEMAPHORES
Thread with 17400 id is executing.....
Thread with 9628 id is executing.....
Thread with 9628 id is executing.....
Thread with 17400 id is executing.....
Thread with 17400 id is executing.....
Thread with 9628 id is executing.....
Thread with 9628 id is executing.....
Thread with 17400 id is executing.....
Thread with 9628 id is executing.....
Thread with 17400 id is executing.....

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>
```