REPORT
OPERATING SYSTEM
ASSIGNMENT SIMULATION BASED

STUDENT'S NAME : T RAM MOHAN.

REGISTRATION N0.:11706425

SEC:EEO34

ROLL NO:36

EMAIL ADDRESS: rammohanthogarchiti@gmail.com

GUIDED BY : PRABHDEEP KAUR.

DESCRIPTION:

A barrier is a tool for synchronising the activity of a number of threads. When a thread reaches a

thread reaches the barrier point, all threads are released and can resume concurrent execution. Assume that the barrier is initialised to N —the number of threads that must wait at the barrier point :

Init(N);

Each thread then performs some work until it reaches the barrier point:

/* do some work for awhile

*/ barrier point();

/* do some work for awhile */

Using sync

• int init(int n) —Initialises the barrier to the specified size.

• int barrier point(void) —Identifies the barrier point. All threads are released from the barrier when the last thread reaches this point.

The return value of each function is used to identify error conditions. Each function will return 0 under normal operation and will return −1 if an error occurs. A testing harness is provided in the source code download to test your implementation of the barrier.

Barriers A barrier is a type of synchronisation method. A barrier for a group of threads or processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier. A barrier is a method to implement

synchronisation. Synchronisation ensures that concurrently executing threads or processes do not execute specific portions of the program at the same time. When a barrier is inserted at a specific point in a program for a group of threads [processes], any thread [process] must stop at this point and cannot proceed until all other threads [processes] reach this barrier.

Complexity: O (n) complexity. “n” is no of thhronisation tools described in this chapter, construct a barrier that implements the follow

QUESTION NO:18

This problem demonstrates the use of semaphores to coordinate three types ofprocesses.6 Santa Claus sleeps in his shop at the North Pole and can only be wakenedby either (1) all nine reindeer being back from their vacation in the South Pacific, or(2) some of the elves having difficulties making toys; to allow Santa to get some sleep,the elves can only wake him when three of them have problems.When three elves arehaving their problems solved, any other elves wishing to visit Santa must wait forthose elves to return. If Santa wakes up to find three elves waiting at his shop’s door,along with the last reindeer having come back from the tropics, Santa has decidedthat the elves can wait until after Christmas, because it is more important to get hissleigh ready. (It is assumed that the reindeer do not want to leave the tropics, andtherefore they stay there until the last possible moment.) The last reindeer to arrivemust get Santa while the others wait in a warming hut before being harnessed to thesleigh. Using synchronization tools like locks, semaphores and monitorsprovide a solution to this problem.

```
#include <pthread.h>

#include <stdlib.h>

#include <assert.h>

#include <unistd.h>

#include <stdio.h>

#include <stdbool.h>

#include <semaphore.h>
```

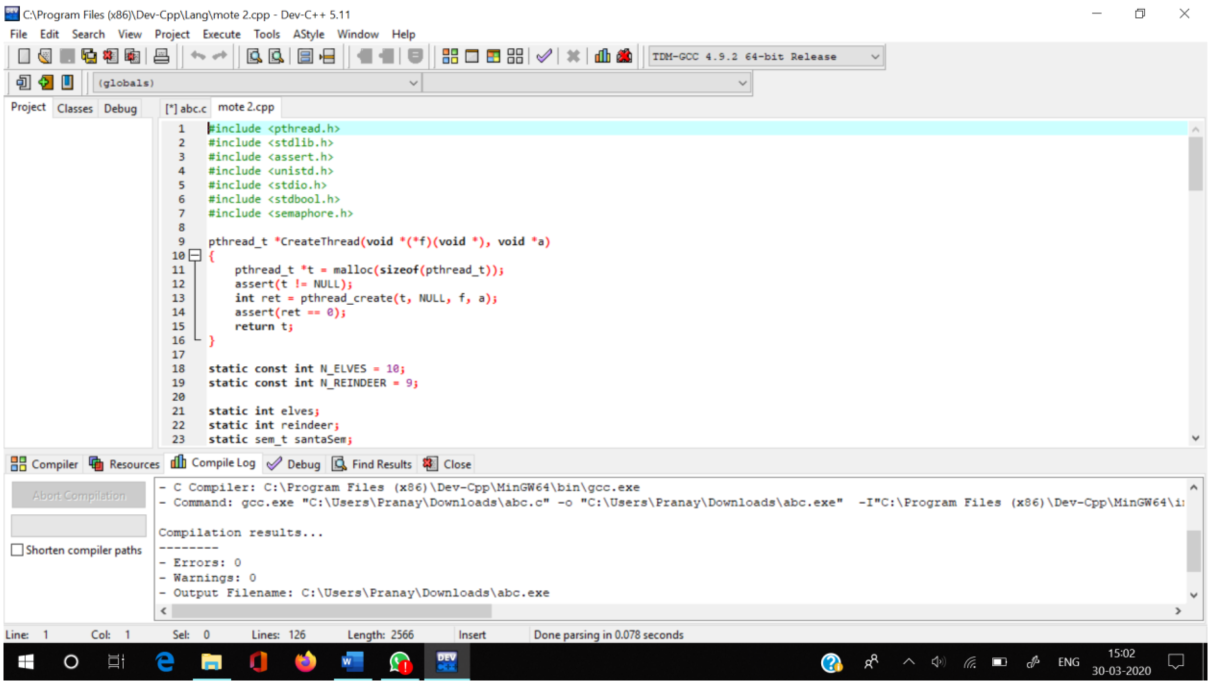
```
pthread_t *CreateThread(void *(*f)(void *), void *a)

{

            pthread_t *t = malloc(sizeof(pthread_t));

            assert(t != NULL);

            int ret = pthread_create(t, NULL, f, a);

            assert(ret == 0);

            return t;

}


static const int N_ELVES = 10;

static const int N_REINDEER = 9;


static int elves;

static int reindeer;

static sem_t santaSem;

static sem_t reindeerSem;

static sem_t elfTex;

static sem_t mutex;


void *SantaClaus(void *arg)

{

            printf("Santa Claus: Hoho, here I am\n");

            while (true)

            {

                        sem_wait(&santaSem);

                        sem_wait(&mutex);
```

```
		if (reindeer == N_REINDEER)
		{
			printf("Santa Claus: preparing sleigh\n");
			for (int r = 0; r < N_REINDEER; r++)
				sem_post(&reindeerSem);
			printf("Santa Claus: make all kids in the world happy\n");
			reindeer = 0;
		}
		else if (elves == 3)
		{
			printf("Santa Claus: helping elves\n");
		}
		sem_post(&mutex);
	}
	return arg;
}

void *Reindeer(void *arg)
{
	int id = (int)arg;
	printf("This is reindeer %d\n", id);
	while (true)
	{
		sem_wait(&mutex);
		reindeer++;
		if (reindeer == N_REINDEER)
			sem_post(&santaSem);
```

```
                sem_post(&mutex);

                sem_wait(&reindeerSem);

                printf("Reindeer %d getting hitched\n", id);

                sleep(20);

        }

        return arg;

}


void *Elve(void *arg)

{

        int id = (int)arg;

        printf("This is elve %d\n", id);

        while (true)

        {

                bool need_help = random() % 100 < 10;

                if (need_help)

                {

                        sem_wait(&elfTex);

                        sem_wait(&mutex);

                        elves++;

                        if (elves == 3)

                                sem_post(&santaSem);

                        else

                                sem_post(&elfTex);

                        sem_post(&mutex);


                        printf("Elve %d will get help from Santa Claus\n", id);
```

```
				sleep(10);


				sem_wait(&mutex);

				elves--;

				if (elves == 0)

					sem_post(&elfTex);

				sem_post(&mutex);

			}

			// Do some work

			printf("Elve %d at work\n", id);

			sleep(2 + random() % 5);

		}

		return arg;

}


int main(int ac, char **av)

{

		elves = 0;

		reindeer = 0;

		sem_init(&santaSem, 0, 0);

		sem_init(&reindeerSem, 0, 0);

		sem_init(&elfTex, 0, 1);

		sem_init(&mutex, 0, 1);


		pthread_t *santa_claus = CreateThread(SantaClaus, 0);


		pthread_t *reindeers[N_REINDEER];
```

```
        for (int r = 0; r < N_REINDEER; r++)

                reindeers[r] = CreateThread(Reindeer, (void *)r + 1);


        pthread_t *elves[N_ELVES];

        for (int e = 0; e < N_ELVES; e++)

                elves[e] = CreateThread(Elve, (void *)e + 1);


        int ret = pthread_join(*santa_claus, NULL);

        assert(ret == 0);

}
```

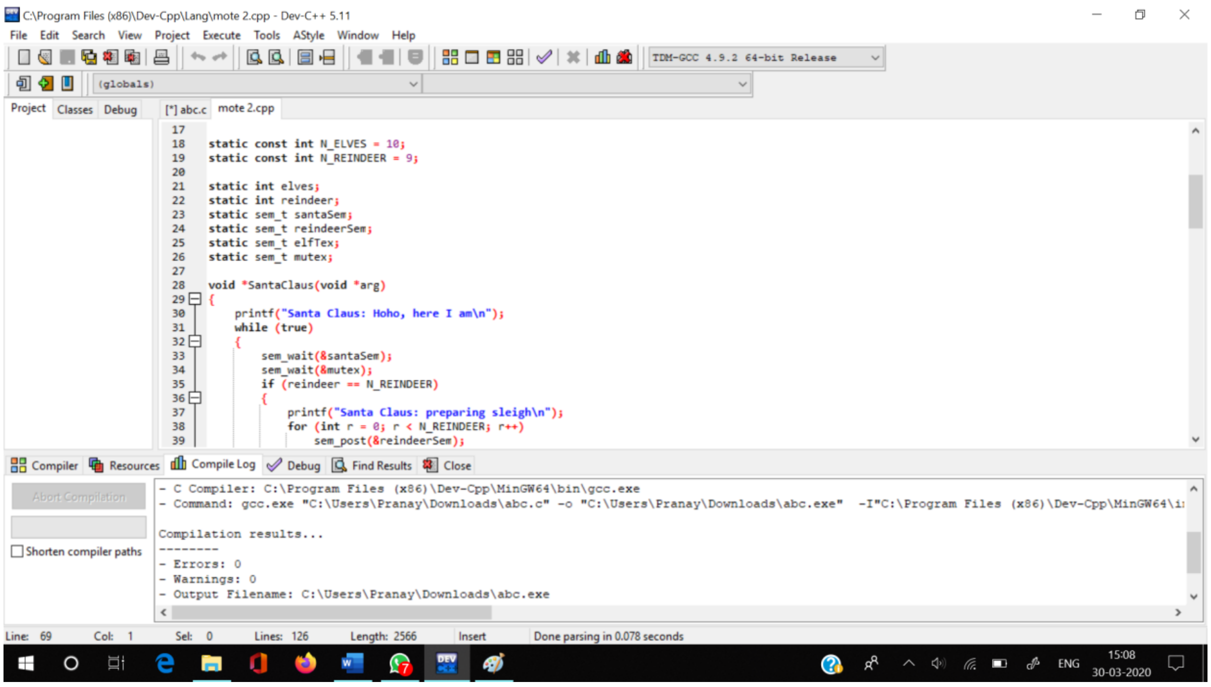


C:\Program Files (x86)\Dev-Cpp\Lang\mote 2.cpp - Dev-C++ 5.11

```
#include <pthread.h>
#include <stdlib.h>
#include <assert.h>
#include <unistd.h>
#include <stdio.h>
#include <stdbool.h>
#include <semaphore.h>

pthread_t *CreateThread(void *(*f)(void *), void *a)
{
    pthread_t *t = malloc(sizeof(pthread_t));
    assert(t != NULL);
    int ret = pthread_create(t, NULL, f, a);
    assert(ret == 0);
    return t;
}

static const int N_ELVES = 10;
static const int N_REINDEER = 9;

static int elves;
static int reindeer;
static sem_t santaSem;
```

```
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "C:\Users\Pranay\Downloads\abc.c" -o "C:\Users\Pranay\Downloads\abc.exe"  -I"C:\Program Files (x86)\Dev-Cpp\MinGW64\i

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pranay\Downloads\abc.exe
```

Line: 1 Col: 1 Sel: 0 Lines: 126 Length: 2566 Insert Done parsing in 0.078 seconds

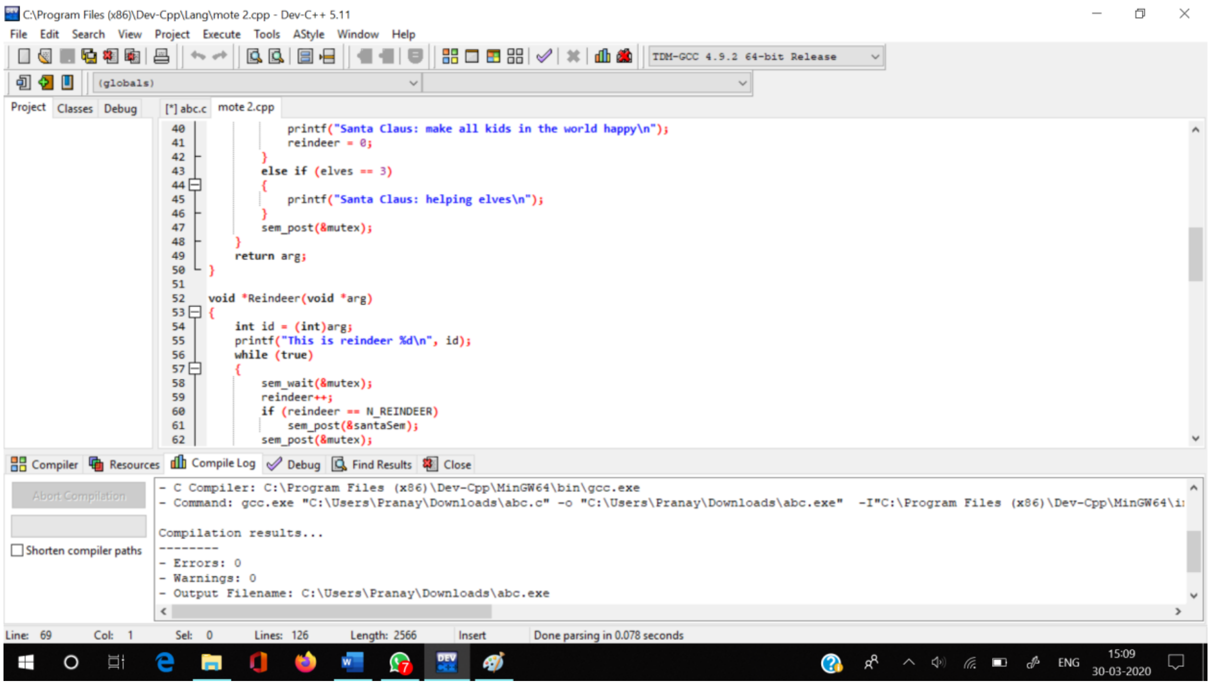

C:\Program Files (x86)\Dev-Cpp\Lang\mote 2.cpp - Dev-C++ 5.11

```c

static const int N_ELVES = 10;
static const int N_REINDEER = 9;

static int elves;
static int reindeer;
static sem_t santaSem;
static sem_t reindeerSem;
static sem_t elfTex;
static sem_t mutex;

void *SantaClaus(void *arg)
{
    printf("Santa Claus: Hoho, here I am\n");
    while (true)
    {
        sem_wait(&santaSem);
        sem_wait(&mutex);
        if (reindeer == N_REINDEER)
        {
            printf("Santa Claus: preparing sleigh\n");
            for (int r = 0; r < N_REINDEER; r++)
                sem_post(&reindeerSem);
```

```
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "C:\Users\Pranay\Downloads\abc.c" -o "C:\Users\Pranay\Downloads\abc.exe"  -I"C:\Program Files (x86)\Dev-Cpp\MinGW64\i

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pranay\Downloads\abc.exe
```

Line: 69 Col: 1 Sel: 0 Lines: 126 Length: 2566 Insert Done parsing in 0.078 seconds

15:08
30-03-2020

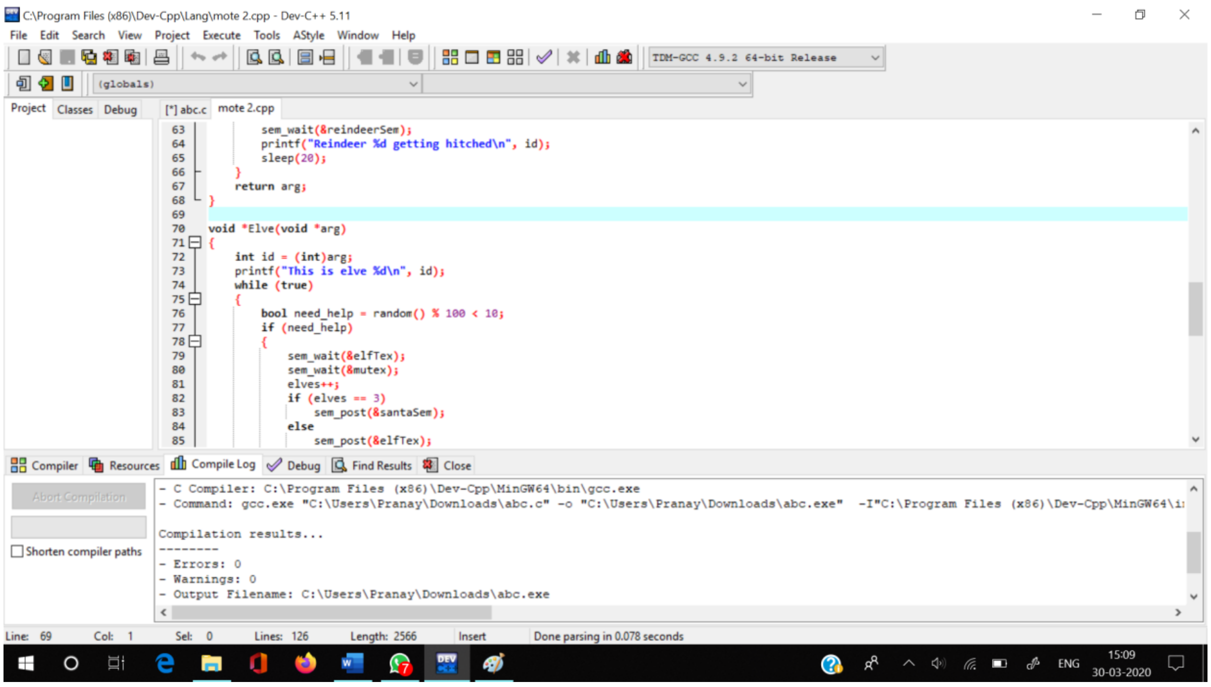

C:\Program Files (x86)\Dev-Cpp\Lang\mote 2.cpp - Dev-C++ 5.11

```c
            printf("Santa Claus: make all kids in the world happy\n");
            reindeer = 0;
        }
        else if (elves == 3)
        {
            printf("Santa Claus: helping elves\n");
        }
        sem_post(&mutex);
    }
    return arg;
}

void *Reindeer(void *arg)
{
    int id = (int)arg;
    printf("This is reindeer %d\n", id);
    while (true)
    {
        sem_wait(&mutex);
        reindeer++;
        if (reindeer == N_REINDEER)
            sem_post(&santaSem);
        sem_post(&mutex);
```

```
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "C:\Users\Pranay\Downloads\abc.c" -o "C:\Users\Pranay\Downloads\abc.exe"  -I"C:\Program Files (x86)\Dev-Cpp\MinGW64\i

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pranay\Downloads\abc.exe
```

Line: 69 Col: 1 Sel: 0 Lines: 126 Length: 2566 Insert Done parsing in 0.078 seconds

15:09
30-03-2020


C:\Program Files (x86)\Dev-Cpp\Lang\mote 2.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] abc.c mote 2.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 69 Col: 1 Sel: 0 Lines: 126 Length: 2566 Insert Done parsing in 0.078 seconds
ENG 15:09 30-03-2020

```
        sem_wait(&reindeerSem);
        printf("Reindeer %d getting hitched\n", id);
        sleep(20);
    }
    return arg;
}

void *Elve(void *arg)
{
    int id = (int)arg;
    printf("This is elve %d\n", id);
    while (true)
    {
        bool need_help = random() % 100 < 10;
        if (need_help)
        {
            sem_wait(&elfTex);
            sem_wait(&mutex);
            elves++;
            if (elves == 3)
                sem_post(&santaSem);
            else
                sem_post(&elfTex);
```

```
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "C:\Users\Pranay\Downloads\abc.c" -o "C:\Users\Pranay\Downloads\abc.exe"  -I"C:\Program Files (x86)\Dev-Cpp\MinGW64\i

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pranay\Downloads\abc.exe
```

ALGORITHM:

This problem demonstrates the use of semaphores to coordinate three types of processes. Santa Claus sleeps in his shop at the North Pole and can only be wakened by either

1) all nine reindeer being back from their vacation in the South Pacific, or

2) by some of the elves having difficulties making toys.

To allow Santa to get some sleep, the elves can only wake him when three of them have problems. While three elves are having their problems solved, any other elves wishing to visit Santa must wait for those elves to return. If Santa wakes up to find three elves waiting at his shop's door, along with the last reindeer having come back from the tropics, Santa has decided that the elves can wait until after Christmas, because it is more important to get his sleigh ready. (It is assumed that the reindeer don't want to leave the tropics, and therefore they stay there until the last possible moment.) The last reindeer to arrive must get Santa while the others wait in a warming hut before being harnessed to the sleigh.

Your team's task is to solve this problem using semaphores. To do this you must write the code for Santa Claus, the reindeers, and the elves. Use whatever semaphores necessary, but be sure that you initialize their values appropriately.

You have the **whole period** to try and solve this problem. During this time you should record your solution for Santa, the reindeers, and the elves. The team(s) judged to have the "best" solution(s) by me can present their solution to the class for review next class.

Santa Outline:

```
while (TRUE) do
    sleep
    if (reindeer at door) then
        tell reindeer to roust other reindeer
        ready sleigh
        deliver toys
    else
        for (each elve) do
            solve problem
            send back to work
        end for
    end if
end while
```

Reindeer Outline:

```
while (TRUE) do
    while(not Dec. 24 and not tired of South Pacific) do
        vacation in South Pacific
    end while
    return to North Pole
    if last reindeer to return then
        wake Santa
    else
        wait in warming hut
    end if
    hook up to sleigh
    fly Santa around world
    return to South Pacific
end while
```

Elf Outline:

```
while (TRUE) do
    while(no problem) do
        build toy
    end while
    problemElfCount++
    if problemElfCount < 3
        wait
    else if problemElfCount = 3 then
        gather other 2 elves with problems
        problemElfCount -= 3
    end if
    if a group of elves off to see Santa
        have your group wait
    end if
    see Santa
    On returning, send another group of elves if there is one waiting
end while
```

THANKING YOU