OPERATING SYSTEMS PRACTICE
(ASSIGNMENT 8)

Name – Mridul Harish
Roll number – CED18I034

Question 1 : Santa Claus Problem

Code :-

```
#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)
{
	static long thread;
	pthread_t* t;
	t = (pthread_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 = r+1)
                    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 = reindeer + 1;
            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 = elves + 1;

			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 = elves - 1;

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

        for (int e = 0; e < N_ELVES; e = e+1)
                elves[e] = CreateThread(Elve, (void *)e + 1);

        int ret = pthread_join(*santa_claus, NULL);
        assert(ret == 0);
}
```

Question 2 : Building H2O problem

Algorithm :-

```
Oxygen code;
mutex.wait()
oxygen += 1
if hydrogen >= 2:
        hydroQueue.signal(2)
        hydrogen -= 2
        oxyQueue.signal()
        oxygen -= 1
else:
        mutex.signal()
        oxyQueue.wait()
        bond()
        barrier.wait()
        mutex.signal()

Hydrogen code;
mutex.wait()
hydrogen += 1
if hydrogen >= 2 and oxygen >= 1:
        hydroQueue.signal(2)
        hydrogen -= 2
        oxyQueue.signal()
        oxygen -= 1
else:
        mutex.signal()
        hydroQueue.wait()
        bond()
```

barrier.wait()

Code :-

```
#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
#include<unistd.h>

sem_t smutex, oxyQueue, hydroQueue;
int oxygen=0; int hydrogen=0;
pthread_t oxyThread, hydroThread1, hydroThread2;

int bond();
void* oxyFn(void* arg);
void* hydroFn(void* arg);

int main()
{
	if(sem_init(&smutex,0,1) == -1)
	{
		perror("error initilalizing semaphore\n");
	}

	if(sem_init(&oxyQueue,0,0) == -1)
	{
		perror("error initilalizing semaphore\n");
	}

	if(sem_init(&hydroQueue,0,0) == -1)
	{
		perror("error initilalizing semaphore\n");
	}

	sleep(2);

	pthread_create(&oxyThread,0,oxyFn, NULL);
	pthread_create(&hydroThread1,0,hydroFn, NULL);
	pthread_create(&hydroThread2,0,hydroFn, NULL);

	for(;;);
}

int bond()
{
```

```
	static int i=0;
	i = i+1;

	if((i%3) == 0)
		printf("** Molecule no. %d created**\n\n",i/3);

	sleep(2);
	return(0);
}

void* oxyFn(void* arg)
{
	while(1)
	{
		sem_wait(&smutex);
		oxygen+=1;

		if(hydrogen >= 2)
		{
			sem_post(&hydroQueue);
			sem_post(&hydroQueue);
			hydrogen-=2;
			sem_post(&oxyQueue);
			oxygen-=1;
		}
		else
		{
			sem_post(&smutex);
		}

		sem_wait(&oxyQueue);
		printf("Oxygen Bond\n");
		bond();
		sleep(3);
		sem_post(&smutex);
		}
}

void* hydroFn(void* arg)
{
	while(1)
	{
		sem_wait(&smutex);
		hydrogen = hydrogen + 1;
```

```
            if(hydrogen >= 2 && oxygen >= 1)
            {
                sem_post(&hydroQueue);
                sem_post(&hydroQueue);
                hydrogen = hydrogen - 2;
                sem_post(&oxyQueue);
                oxygen = oxygen - 1;
            }
            else
            {
                sem_post(&smutex);
            }

            sem_wait(&hydroQueue);

        printf("Hydrogen Bond\n");
        bond();
        sleep(3);
      }
}
```

Output :