ASSIGNMENT-7

COE18B023
G. Hruthik

1. Simulate the Producer Consumer code discussed in the class.
CODE:

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#define MAX 5
#define BUFFERSIZE 2
sem_t empty;
sem_t full;
int in = 0;
int out = 0;
int buffer[BUFFERSIZE];
pthread_mutex_t mutex;
void *producer(void *param)
{
int item;
for (int i = 0; i < MAX; i++)
{
item = rand() % 100;
sem_wait(&empty);
pthread_mutex_lock(&mutex);
buffer[in] = item;
printf("Producer produces Item %d at position : %d\n", buffer[in], in);
in = (in + 1) % BUFFERSIZE;
pthread_mutex_unlock(&mutex);
sem_post(&full);
}
}
```

```
void *consumer(void *param)
{
for (int i = 0; i < MAX; i++)
{
sem_wait(&full);pthread_mutex_lock(&mutex);
int item = buffer[out];
printf("Consumer consumes Item %d at position : %d\n", item, out);
out = (out + 1) % BUFFERSIZE;
pthread_mutex_unlock(&mutex);
sem_post(&empty);
}
}
int main()
{
pthread_t tid[2];
pthread_mutex_init(&mutex, NULL);
sem_init(&empty, 0, BUFFERSIZE);
sem_init(&full, 0, 0);
pthread_create(&tid[0], NULL, producer, NULL);
pthread_create(&tid[1], NULL, consumer, NULL);
pthread_join(tid[0], NULL);
pthread_join(tid[1], NULL);
pthread_mutex_destroy(&mutex);
sem_destroy(&empty);
sem_destroy(&full);
return 0;
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ gcc q1.c -lpthread
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Producer produces Item 83 at position : 0
Producer produces Item 86 at position : 1
Consumer consumes Item 83 at position : 0
Consumer consumes Item 86 at position : 1
Producer produces Item 77 at position : 0
Producer produces Item 15 at position : 1
Consumer consumes Item 77 at position : 0
Consumer consumes Item 15 at position : 1
Producer produces Item 93 at position : 0
Consumer consumes Item 93 at position : 0
```

2. Extend the producer consumer simulation in Q1 to sync access of critical
data using Petersonsalgorithm.

CODE:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
int item;
int flag[2];
int turn;
sem_t empty, full;
void lock_ini()
{
flag[0] = 0;
flag[1] = 0;
turn = 0;
}
void lock(int index)
{
flag[index] = 1;
turn = 1 - index;
while (flag[1 - index] == 1 && turn == 1 - index);
}
```

```
void unlock(int index)
{
flag[index] = 0;
}
void *producer()
{
int i;
printf("\nPRODUCER THREAD ID: %ld\n", pthread_self());
for (i = 1; i < 6; i++)
{
sem_wait(&empty);
lock(0);
//entry-section
item = i;
//critical section
unlock(0);
//critical section
sem_post(&full);
//exit section
printf("Produced item: %d\n", item);
}
}
void *consumer()
{
int i, total = 0;printf("CONSUMER THREAD ID: %ld\n", pthread_self());
for (i = 1; i < 6; i++)
{
sem_wait(&full);
lock(1);
//entry-section
total = total + item; //critical section
unlock(1);
//critical section
sem_post(&empty); //exit section
printf("Consumed item: %d\n", item);
```

```
}
}
int main()
{
pthread_t tid[2];
lock_ini();
sem_init(&empty, 0, 1);
sem_init(&full, 0, 0);
pthread_create(&tid[0], NULL, producer, NULL);
pthread_create(&tid[1], NULL, consumer, NULL);
pthread_join(tid[0], NULL);
pthread_join(tid[1], NULL);
printf("\n");
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ gcc q2.c -lpthread
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out

PRODUCER THREAD ID: 140389515888384
Produced item: 1
CONSUMER THREAD ID: 140389507495680
Consumed item: 1
Produced item: 2
Consumed item: 2
Produced item: 3
Consumed item: 3
Consumed item: 4
Produced item: 4
Produced item: 5
Consumed item: 5
```

3. Dictionary Problem: Let the producer set up a dictionary of at least 20 words
with three attributes
(Word, Primary meaning, Secondary meaning) and let the consumer search forthe word and retrieve
its respective primary and secondary meaning.
Note: This can be implemented using either Mutex locks or Peterson's

algorithm.

CODE:

```
#include <pthread.h>
#include <semaphore.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
sem_t wrt;
char search[100];
pthread_mutex_t mutex;
int numreader = 0;
int k=0;
int flag;
typedef struct
{
char word[100];
char primary[1000];
char secondary[100];
} dict;
dict dictionary[20];
void *writer(void *wno)
{
sem_wait(&wrt);
printf("Enter the word:\t");
scanf("%s",search);
int i;
strcpy(dictionary[k].word,search);
printf("Enter meaning:\t");
scanf("%s",dictionary[k].primary);
printf("Enter secondary meaning:\t");
scanf("%s",dictionary[k].secondary);
printf("%s is added to the dictionary.\n\n",dictionary[k].word);
k++;
sem_post(&wrt);
```

```
}
void *reader(void *rno)
{
pthread_mutex_lock(&mutex);
numreader++;
if(numreader == 1) {sem_wait(&wrt);
}
pthread_mutex_unlock(&mutex);
int i;
printf("Enter word to search:\t");
scanf("%s",search);
for(i=0;i<20;i++)
{
if (strcmp(search,dictionary[i].word)==0)
{
flag=1;
printf("Primary Meaning:\t%s\n", dictionary[i].primary);
printf("Secondary Meaning:\t%s\n", dictionary[i].secondary);
exit(0);
}
}
if(flag==0)
printf("Given word is not found\n");
pthread_mutex_lock(&mutex);
numreader--;
if(numreader == 0)
{
sem_post(&wrt);
}
pthread_mutex_unlock(&mutex);
}
int main()
{
pthread_t read,write[20];
pthread_mutex_init(&mutex, NULL);
```

```
sem_init(&wrt,0,1);
for(int i = 0; i < 20; i++)
pthread_create(&write[i], NULL, (void *)writer, NULL);
sleep(1);
pthread_create(&read, NULL, (void *)reader, NULL);
pthread_join(read, NULL);
for(int i = 0; i < 20; i++)
pthread_join(write[i], NULL);
pthread_mutex_destroy(&mutex);
sem_destroy(&wrt);
return 0;
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ gcc q3.c -lpthread
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Enter the word: 1
Enter meaning:  one
Enter secondary meaning:        ek
1 is added to the dictionary.

Enter the word: 2
Enter meaning:  two
Enter secondary meaning:        dhoo
2 is added to the dictionary.

Enter the word: 3
Enter meaning:  three
Enter secondary meaning:        theen
3 is added to the dictionary.

Enter the word: 4
Enter meaning:  four
Enter secondary meaning:        char
4 is added to the dictionary.

Enter the word: 5
Enter meaning:  five
Enter secondary meaning:        panch
5 is added to the dictionary.

Enter the word: 6
Enter meaning:  six
Enter secondary meaning:        chey
6 is added to the dictionary.

Enter the word: 7
Enter meaning:  seven
Enter secondary meaning:        saath
7 is added to the dictionary.

Enter the word: 8
Enter meaning:  eight
Enter secondary meaning:        aat
8 is added to the dictionary.

Enter the word: 9
Enter meaning:  nine
Enter secondary meaning:        nov
9 is added to the dictionary.

Enter the word: 10
Enter meaning:  ten
Enter secondary meaning:        dhas
10 is added to the dictionary.

Enter the word: 11
Enter meaning:  eleven
Enter secondary meaning:        kouh
```

```
hruthik@hruthik-dell-Vostro: ~/Desktop/OS/LAB7

Enter the word: 11
Enter meaning:  eleven
Enter secondary meaning:        kouh
11 is added to the dictionary.

Enter the word: 12
Enter meaning:  twelve
Enter secondary meaning:        dtg
12 is added to the dictionary.

Enter the word: 13
Enter meaning:  thirteen
Enter secondary meaning:        dtg
13 is added to the dictionary.

Enter the word: 14 fourteen
Enter meaning:  Enter secondary meaning:        sdc
14 is added to the dictionary.

Enter the word: 15
Enter meaning:  fifteen
Enter secondary meaning:        yuy
15 is added to the dictionary.

Enter the word: 16
Enter meaning:  sixteen
Enter secondary meaning:        cyt
16 is added to the dictionary.

Enter the word: 17
Enter meaning:  seventeen
Enter secondary meaning:        htf
17 is added to the dictionary.

Enter the word: 18
Enter meaning:  eighteen
Enter secondary meaning:
f
18 is added to the dictionary.

Enter the word: 19
Enter meaning:  nineteen
Enter secondary meaning:        tfc
19 is added to the dictionary.

Enter the word: 20
Enter meaning:  twenty
Enter secondary meaning:        thxr
20 is added to the dictionary.

Enter word to search:   12
Primary Meaning:        twelve
Secondary Meaning:      dtg
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$
```

Non-Mandatory (Extra credits):

4. Extend Q3 to avoid duplication of dictionary entries and implement an

efficient binary search on the consumer side in a multithreaded fashion.

CODE:

```
#include <pthread.h>
#include <semaphore.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
sem_t wrt;
char search[100];pthread_mutex_t mutex;
int numreader = 0;
int k=0;
int flag;
typedef struct
{
char word[100];
char primary[100];
char secondary[100];
} dict;
dict dictionary[20];
void *writer(void *wno)
{
sem_wait(&wrt);
flag=1;
printf("Enter word:\t");
scanf("%s",search);
int i;
for(i=0;i<20;i++)//duplicates
{
if(strcmp(dictionary[i].word,search)==0)
{
printf("WORD ALREADY EXISTS\n");
flag=0;
```

```
break;
}
}
if(flag==1)
{
strcpy(dictionary[k].word,search);
printf("Enter primary meaning:\t");
scanf("%s",dictionary[k].primary);
printf("Enter secondary meaning:\t");
scanf("%s",dictionary[k].secondary);
printf("%s is added to the dictionary.\n\n",dictionary[k].word);
k++;
}
sem_post(&wrt);
}
void *reader(void *rno)
{
pthread_mutex_lock(&mutex);
numreader++;
if(numreader == 1) {sem_wait(&wrt);
}
pthread_mutex_unlock(&mutex);
int i,flag=0,low=0,high=19;
printf("Enter word to search:\t");
scanf("%s",search);
while(low<=high)
{
int mid=(low+high)/2;
if (strcmp(search,dictionary[mid].word)==0)
{
printf("Primary Meaning: %s\n", dictionary[mid].primary);
printf("Secondary Meaning: %s\n", dictionary[mid].secondary);
exit(0);
}
else if(strcmp(search,dictionary[mid].word)>0)
```

```
{
high=high;
low=mid+1;
}
else
{
low=low;
high=mid-1;
}
}
printf("Given word is not found\n");
pthread_mutex_lock(&mutex);
numreader--;
if(numreader == 0)
{
sem_post(&wrt);
}
pthread_mutex_unlock(&mutex);
}
int main()
{
pthread_t read,write[20];
pthread_mutex_init(&mutex, NULL);
sem_init(&wrt,0,1);
for(int i = 0; i < 20; i++)
pthread_create(&write[i], NULL, (void *)writer, NULL);
sleep(1);
pthread_create(&read, NULL, (void *)reader, NULL);
pthread_join(read, NULL);for(int i = 0; i < 20; i++)
pthread_join(write[i], NULL);
pthread_mutex_destroy(&mutex);
sem_destroy(&wrt);
return 0;
}
```

OUTPUT:

hruthik@hruthik-dell-Vostro: ~/Desktop/OS/LAB7

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ gcc q4.c -lpthread
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Enter word:     1
Enter primary meaning:  one
Enter secondary meaning:        ef
1 is added to the dictionary.

Enter word:     2
Enter primary meaning:  two
Enter secondary meaning:        fg
2 is added to the dictionary.

Enter word:     3
Enter primary meaning:  three
Enter secondary meaning:        dfv
3 is added to the dictionary.

Enter word:     4
Enter primary meaning:  gb
Enter secondary meaning:        fb
4 is added to the dictionary.

Enter word:     5
Enter primary meaning:  gtg
Enter secondary meaning:        sef
5 is added to the dictionary.

Enter word:     6
Enter primary meaning:  itug
Enter secondary meaning:        sfbt
6 is added to the dictionary.

Enter word:     7
Enter primary meaning:  gdth
Enter secondary meaning:        rgr
7 is added to the dictionary.

Enter word:     8
Enter primary meaning:  sdcg
Enter secondary meaning:        sfvdf
8 is added to the dictionary.

Enter word:     9
Enter primary meaning:  sfv
Enter secondary meaning:        bhfd
9 is added to the dictionary.

Enter word:     10
Enter primary meaning:  afnr
Enter secondary meaning:        sbhy
10 is added to the dictionary.

Enter word:     11
Enter primary meaning:  gbbds
Enter secondary meaning:        rgrt
```

```
hruthik@hruthik-dell-Vostro: ~/Desktop/OS/LAB7
10 is added to the dictionary.

Enter word:     11
Enter primary meaning:  gbbds
Enter secondary meaning:        rgrt
11 is added to the dictionary.

Enter word:     12
Enter primary meaning:  srgrt
Enter secondary meaning:        dgbds
12 is added to the dictionary.

Enter word:     13
Enter primary meaning:  hnbrf
Enter secondary meaning:        dvry
13 is added to the dictionary.

Enter word:     14
Enter primary meaning:  sdbgbry
Enter secondary meaning:        dbsdgb
14 is added to the dictionary.

Enter word:     15
Enter primary meaning:  dsbbynyg
Enter secondary meaning:        c vbr
15 is added to the dictionary.

Enter word:     Enter primary meaning:  16
Enter secondary meaning:        egrreb
vbr is added to the dictionary.

Enter word:     t17
Enter primary meaning:  erge
Enter secondary meaning:        ergehy
t17 is added to the dictionary.

Enter word:     18
Enter primary meaning:  jyfytd
Enter secondary meaning:        ougt
18 is added to the dictionary.

Enter word:     19
Enter primary meaning:  utfrd
Enter secondary meaning:        yfyrdh
19 is added to the dictionary.

Enter word:     20
Enter primary meaning:  jyfrty
Enter secondary meaning:        yfydtr
20 is added to the dictionary.

Enter word to search:   12
Primary Meaning: srgrt
Secondary Meaning: dgbds
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$
```

(A) Implement the Dining Philosophers and Reader Writer Problem of Synchronization (test drive the codes discussed in the class)

DINING PHILOSOPHERS:

CODE:

```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#include<unistd.h>
#define N 5
#define THINKING 0
#define HUNGRY 1#define EATING 2
#define LEFT (ph_num+4)%N
#define RIGHT (ph_num+1)%N
sem_t mutex;
sem_t S[N];
void * philospher(void *num);
void take_fork(int);
void put_fork(int);
void test(int);
int state[N];
int phil_num[N]={0,1,2,3,4};
int main()
{
int i;
pthread_t thread_id[N];
sem_init(&mutex,0,1);
for(i=0;i<N;i++)
sem_init(&S[i],0,0);
for(i=0;i<N;i++)
{
pthread_create(&thread_id[i],NULL,philospher,&phil_num[i]);
printf("Philosopher %d is thinking\n",i+1);
}
for(i=0;i<N;i++)
pthread_join(thread_id[i],NULL);
}
void *philospher(void *num)
{
```

```
while(1)
{
int *i = num;
sleep(1);
take_fork(*i);
sleep(0);
put_fork(*i);
}
}
void take_fork(int ph_num)
{
sem_wait(&mutex);
state[ph_num] = HUNGRY;
printf("Philosopher %d is Hungry\n",ph_num+1);
test(ph_num); sem_post(&mutex);
sem_wait(&S[ph_num]);
sleep(1);
}void test(int ph_num)
{
if (state[ph_num] == HUNGRY && state[LEFT] != EATING &&
state[RIGHT] !=
EATING) {
state[ph_num] = EATING;
sleep(2);
printf("Philosopher %d takes fork %d and %d\n",ph_num+1,LEFT+1,
ph_num+1);
printf("Philosopher %d is Eating\n",ph_num+1);
sem_post(&S[ph_num]);
}
}
void put_fork(int ph_num)
{
sem_wait(&mutex);
state[ph_num] =THINKING;
printf("Philosopher %d putting fork %d and %d down\n",
```

```
ph_num+1,LEFT+1,ph_num+1);
printf("Philosopher %d is
thinking\n",ph_num+1);
test(LEFT);
test(RIGHT);
sem_post(&mutex);
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ gcc Dining_Philosophers.c -lpthread
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 1 is Hungry
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 2 is Hungry
Philosopher 4 is Hungry
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 3 is Hungry
Philosopher 5 is Hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 1 is Hungry
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 4 is Hungry
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 2 is Hungry
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 5 is Hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 3 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 1 is Hungry
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 3 takes fork 2 and 3
```

READER_WRITERS:

CODE:

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

```
#include<unistd.h>
sem_t mutex, writeblock;
int data = 0,rcount = 0;
void *reader(void *arg)
{
int f;
f= (int)arg;
sem_wait(&mutex);
rcount = rcount + 1;
if(rcount==1)
sem_wait(&writeblock);
sem_post(&mutex);
printf("Data read by the reader%d is %d\n",f,data);
sleep(1);
sem_wait(&mutex);
rcount = rcount - 1;
if(rcount==0)
sem_post(&writeblock);
sem_post(&mutex);
}
void *writer(void *arg)
{
int f;
f = (int) arg;
sem_wait(&writeblock);
data++;
printf("Data writen by the writer%d is %d\n",f,data);
sleep(1);
sem_post(&writeblock);
}
int main()
{
int i,b;
pthread_t rtid[5],wtid[5];
sem_init(&mutex,0,1);
```

```
sem_init(&writeblock,0,1);
for(i=0;i<=2;i++)
{
pthread_create(&wtid[i],NULL, writer, (void *)i);
pthread_create(&rtid[i],NULL,reader, (void *)i);
}
for(i=0;i<=2;i++)
{pthread_join(wtid[i],NULL);
pthread_join(rtid[i],NULL);
}
return 0;
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Data writen by the writer0 is 1
Data read by the reader0 is 1
Data read by the reader2 is 1
Data read by the reader1 is 1
Data writen by the writer1 is 2
Data writen by the writer2 is 3
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$
```

(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)
{
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);
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Santa Claus: Hoho, here I am
This is reindeer 2
This is reindeer 3
This is reindeer 1
This is reindeer 4
This is reindeer 5
This is reindeer 6
This is reindeer 7
This is reindeer 8
This is reindeer 9
Santa Claus: preparing sleigh
Reindeer 3 getting hitched
Reindeer 1 getting hitched
Reindeer 2 getting hitched
Reindeer 4 getting hitched
Reindeer 9 getting hitched
This is elve 3
Santa Claus: make all kids in the world happy
This is elve 4
Elve 4 at work
Reindeer 7 getting hitched
Reindeer 5 getting hitched
This is elve 2
Elve 2 at work
Elve 3 at work
This is elve 7
Elve 7 at work
This is elve 8
Elve 8 at work
Reindeer 8 getting hitched
This is elve 5
This is elve 9
Reindeer 6 getting hitched
This is elve 1
Elve 1 at work
This is elve 10
Elve 10 at work
Elve 9 at work
This is elve 6
Elve 6 at work
Elve 5 at work
Elve 3 at work
Elve 9 at work
Elve 8 at work
Elve 10 at work
Elve 5 at work
Elve 4 at work
Elve 7 at work
Elve 6 at work
Elve 2 at work
Elve 8 at work
Elve 5 at work
```

(2) H2O Problem :

CODE:

```
#include <pthread.h>
#include <stdio.h>
#include <semaphore.h>
#include <unistd.h>
sem_t smutex,oxyQueue,hydroQueue;
int oxygen=0,hydregen=0;
pthread_t oxyThread,hydroThread1,hydroThread2;
int bond(){
static int i=0;
i++;
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(hydregen>=2){
sem_post(&hydroQueue);
sem_post(&hydroQueue);
hydregen-=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);
hydregen+=1;
if(hydregen>=2 && oxygen>=1){
sem_post(&hydroQueue);
sem_post(&hydroQueue);
hydregen-=2;
sem_post(&oxyQueue);
oxygen-=1;
}
else{
sem_post(&smutex);
}
sem_wait(&hydroQueue);
printf("Hydrogen Bond\n");
bond();
sleep(3);
}
}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(;;);
}
```

OUTPUT:

```
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ gcc H2O.c -lpthread
hruthik@hruthik-dell-Vostro:~/Desktop/OS/LAB7$ ./a.out
Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 1 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 2 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 3 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 4 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 5 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 6 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 7 created**

Hydrogen Bond
Hydrogen Bond
Oxygen Bond
** Molecule no. 8 created**
```

----------THE END--------