Synchronization - II

COM301-P Assignment - 8

G S Santhosh Raghul
COE18B045

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

Reader-Writer Problem :

code :

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

#define NO WRITERS 5
#define NO READERS 10

sem t mutex,writeblock;
int data=0,rcount=0;

void *reader(void *args); // each reader thread reads the data once and displays
void *writer(void *args); // each writer thread increments data by 1

int main()
{
   srand(time(0));
   pthread t rtid[NO READERS], wtid[NO WRITERS];
   pthread attr t rattr[NO READERS],wattr[NO WRITERS];
   int a[NO READERS];
   sem init(&mutex,0,1);
   sem init(&writeblock,0,1);
   for(int i=0;i<NO READERS;i++)
   {
       a[i]=i;
```

```
        if(i<NO WRITERS) // create NO WRITERS no of r,w threads
        {
            pthread attr init(&wattr[i]);
            pthread create(&wtid[i],NULL,writer,(void *)&a[i]);
            pthread attr init(&rattr[i]);
            pthread create(&rtid[i],NULL,reader,(void *)&a[i]);
        }
        else // just to create some randomness, create the rest of the read threads after sleeping for a random amount of time
        {
            sleep(rand()%5);
            pthread attr init(&rattr[i]);
            pthread create(&rtid[i],NULL,reader,(void *)&a[i]);
        }
    }
    for(int i=0;i<NO WRITERS;i++)
        pthread join(wtid[i],NULL);
    for(int i=0;i<NO READERS;i++)
        pthread join(rtid[i],NULL);
    return 0;
}

void *reader(void *args)
{
    int f;
    f = *((int*)args);
    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);
    srand(time(0)*f);
    sleep(1);
    sem wait(&mutex);
    rcount = rcount - 1;
    if(rcount == 0)
        sem post(&writeblock);
    sem post(&mutex);
}
```

```c
void *writer(void *args)
{
   int f,t;
   f=*((int*)args);
   sleep(rand()%5); // sleeps for a random amount of time
   sem wait(&writeblock);
   data++;
   printf("Data written by the writer %d is %d\n",f,data);
   sem post(&writeblock);
}
```

output :

```
santi@edith:~/OS/L8$ gcc read-writ
e.c -pthread
santi@edith:~/OS/L8$ ./a.out
Data written by the writer 1 is 1
Data read by the reader 0 is 1
Data read by the reader 2 is 1
Data read by the reader 1 is 1
Data read by the reader 4 is 1
Data read by the reader 3 is 1
Data read by the reader 5 is 1
Data written by the writer 2 is 2
Data written by the writer 3 is 3
Data read by the reader 6 is 3
Data read by the reader 7 is 3
Data written by the writer 0 is 4
Data written by the writer 4 is 5
Data read by the reader 8 is 5
Data read by the reader 9 is 5
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./a.out
Data read by the reader 0 is 0
Data read by the reader 1 is 0
Data read by the reader 2 is 0
Data read by the reader 3 is 0
Data read by the reader 4 is 0
Data read by the reader 5 is 0
Data written by the writer 3 is 1
Data written by the writer 0 is 2
Data written by the writer 1 is 3
Data written by the writer 2 is 4
Data written by the writer 4 is 5
Data read by the reader 6 is 5
Data read by the reader 7 is 5
Data read by the reader 8 is 5
Data read by the reader 9 is 5
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./a.out
Data read by the reader 1 is 0
Data read by the reader 2 is 0
Data read by the reader 3 is 0
Data read by the reader 4 is 0
Data read by the reader 0 is 0
Data written by the writer 3 is 1
Data read by the reader 5 is 1
Data written by the writer 1 is 2
Data written by the writer 0 is 3
Data written by the writer 2 is 4
Data written by the writer 4 is 5
Data read by the reader 6 is 5
Data read by the reader 7 is 5
Data read by the reader 8 is 5
Data read by the reader 9 is 5
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./a.out
Data read by the reader 0 is 0
Data read by the reader 1 is 0
Data read by the reader 2 is 0
Data read by the reader 3 is 0
Data read by the reader 4 is 0
Data written by the writer 2 is 1
Data written by the writer 3 is 2
Data read by the reader 5 is 2
Data written by the writer 4 is 3
Data written by the writer 1 is 4
Data written by the writer 0 is 5
Data read by the reader 6 is 5
Data read by the reader 7 is 5
Data read by the reader 8 is 5
Data read by the reader 9 is 5
santi@edith:~/OS/L8$
```

Dining Philosophers Problem :

code :

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

#define N 5
#define THINKING 0
#define HUNGRY 1
#define EATING 2
#define LEFT (ph num+N-1)%N
#define RIGHT (ph num+1)%N

sem t mutex,S[N];
int state[N];

void* philosopher runner(void*);
void take chopstick(int);
void put chopstick(int);
void test(int);
void int sig handler(int);

int main()
{
   signal(SIGINT,int sig handler);
   int i,phil num[N];
   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++)
    {
        phil num[i]=i;
        pthread create(&thread id[i],NULL,philosopher runner,&phil num[i]);
        printf("philosopher %d is thinking\n",i+1);
    }

    for(i=0;i<N;i++)
        pthread join(thread id[i],NULL);

    return 0;
}

void* philosopher runner(void* num)
{
    while(1)
    {
        int *i=num;
        sleep(1);
        take chopstick(*i);
        sleep(0);
        put chopstick(*i);
    }
}


void take chopstick(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 is taking chopsticks %d and %d\n",ph num+1,ph num+1,LEFT+1);
       printf("philosopher %d is eating\n",ph num+1);
       sem post(&S[ph num]);
   }
}

void put chopstick(int ph num)
{
   sem wait(&mutex);
   state[ph num]=THINKING;
   printf("philosopher %d is putting chopsticks %d and %d down\n",ph num+1,ph num+1,LEFT+1);
   printf("philosopher %d is thinking\n",ph num+1);
   test(LEFT);
   test(RIGHT);
   sem post(&mutex);
}

void int sig handler(int signum)
{
   printf("\nexiting...\n");
   exit(0);
}
```

output in next page...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash, bash, bash

```
santi@edith:~/OS/L8$ gcc dining-philosophers.c -pthread
santi@edith:~/OS/L8$ ./a.out
philosopher 1 is thinking
philosopher 2 is thinking
philosopher 3 is thinking
philosopher 4 is thinking
philosopher 5 is thinking
philosopher 2 is hungry
philosopher 2 is taking chopsticks 2 and 1
philosopher 2 is eating
philosopher 1 is hungry
philosopher 4 is hungry
philosopher 4 is taking chopsticks 4 and 3
philosopher 4 is eating
philosopher 5 is hungry
philosopher 3 is hungry
philosopher 2 is putting chopsticks 2 and 1 down
philosopher 2 is thinking
philosopher 1 is taking chopsticks 1 and 5
philosopher 1 is eating
philosopher 4 is putting chopsticks 4 and 3 down
philosopher 4 is thinking
philosopher 3 is taking chopsticks 3 and 2
philosopher 3 is eating
philosopher 2 is hungry
philosopher 1 is putting chopsticks 1 and 5 down
philosopher 1 is thinking
philosopher 5 is taking chopsticks 5 and 4
philosopher 5 is eating
philosopher 4 is hungry
philosopher 3 is putting chopsticks 3 and 2 down
philosopher 3 is thinking
philosopher 2 is taking chopsticks 2 and 1
philosopher 2 is eating
philosopher 1 is hungry
philosopher 5 is putting chopsticks 5 and 4 down
philosopher 5 is thinking
philosopher 4 is taking chopsticks 4 and 3
philosopher 4 is eating
philosopher 3 is hungry
philosopher 2 is putting chopsticks 2 and 1 down
philosopher 2 is thinking
philosopher 1 is taking chopsticks 1 and 5
philosopher 1 is eating
philosopher 5 is hungry
philosopher 4 is putting chopsticks 4 and 3 down
philosopher 4 is thinking
^C
exiting...
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./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 is taking chopsticks 1 and 5
philosopher 1 is eating
philosopher 2 is hungry
philosopher 3 is hungry
philosopher 3 is taking chopsticks 3 and 2
philosopher 3 is eating
philosopher 4 is hungry
philosopher 5 is hungry
philosopher 1 is putting chopsticks 1 and 5 down
philosopher 1 is thinking
philosopher 5 is taking chopsticks 5 and 4
philosopher 5 is eating
philosopher 3 is putting chopsticks 3 and 2 down
philosopher 3 is thinking
philosopher 2 is taking chopsticks 2 and 1
philosopher 2 is eating
philosopher 1 is hungry
philosopher 5 is putting chopsticks 5 and 4 down
philosopher 5 is thinking
philosopher 4 is taking chopsticks 4 and 3
philosopher 4 is eating
philosopher 3 is hungry
philosopher 2 is putting chopsticks 2 and 1 down
philosopher 2 is thinking
philosopher 1 is taking chopsticks 1 and 5
philosopher 1 is eating
philosopher 5 is hungry
philosopher 4 is putting chopsticks 4 and 3 down
philosopher 4 is thinking
philosopher 3 is taking chopsticks 3 and 2
philosopher 3 is eating
philosopher 2 is hungry
philosopher 1 is putting chopsticks 1 and 5 down
philosopher 1 is thinking
philosopher 5 is taking chopsticks 5 and 4
philosopher 5 is eating
philosopher 4 is hungry
philosopher 3 is putting chopsticks 3 and 2 down
philosopher 3 is thinking
^C
exiting...
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./a.out
philosopher 1 is thinking
philosopher 2 is thinking
philosopher 3 is thinking
philosopher 4 is thinking
philosopher 5 is thinking
philosopher 2 is hungry
philosopher 2 is taking chopsticks 2 and 1
philosopher 2 is eating
philosopher 1 is hungry
philosopher 5 is hungry
philosopher 5 is taking chopsticks 5 and 4
philosopher 5 is eating
philosopher 4 is hungry
philosopher 3 is hungry
philosopher 2 is putting chopsticks 2 and 1 down
philosopher 2 is thinking
philosopher 3 is taking chopsticks 3 and 2
philosopher 3 is eating
philosopher 5 is putting chopsticks 5 and 4 down
philosopher 5 is thinking
philosopher 1 is taking chopsticks 1 and 5
philosopher 1 is eating
philosopher 2 is hungry
philosopher 3 is putting chopsticks 3 and 2 down
philosopher 3 is thinking
philosopher 4 is taking chopsticks 4 and 3
philosopher 4 is eating
philosopher 5 is hungry
philosopher 1 is putting chopsticks 1 and 5 down
philosopher 1 is thinking
philosopher 2 is taking chopsticks 2 and 1
philosopher 2 is eating
philosopher 3 is hungry
philosopher 4 is putting chopsticks 4 and 3 down
philosopher 4 is thinking
philosopher 5 is taking chopsticks 5 and 4
philosopher 5 is eating
philosopher 1 is hungry
philosopher 2 is putting chopsticks 2 and 1 down
philosopher 2 is thinking
philosopher 3 is taking chopsticks 3 and 2
philosopher 3 is eating
philosopher 4 is hungry
philosopher 5 is putting chopsticks 5 and 4 down
philosopher 5 is thinking
^C
exiting...
santi@edith:~/OS/L8$
```

(B) Choose any 2 of the following problems whose details are available in the Downy Book on Semaphores (attached) and implement semaphores based solutions to the same.

(1) Santa Claus Problem
(2) H2O Problem
(3) Baboon Crossing Problem
(4) Dining Hall Problem
(5) Senate Bus Problem

(1) Santa Claus Problem

explanation :

Santa - first check if 9 reindeers have come. If yes, Santa has to prepare the sleigh and hitch all the reindeers one by one. If no, check if there are 3 elves (because reindeers have more priority). If yes, Santa has to help the elves. This will go on infinitely.

Reindeers - when the 9th reindeer comes, it has to signal Santa to get the sleigh. Then, each reindeer has to wait for Santa's signal and get hitched.

Elves - first 2 elves just signal the elf semaphore and wait. The 3rd elf does not signal the elf semaphore right away. Third elf has to signal Santa semaphore to wake Santa up and then wait until other 2 elves get help and finally release the elf semaphore.

code :

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

int elves=0,reindeer=0;
sem_t mutex;
sem_t santa_sem,reindeer_sem,elf_sem;
```

```
void* santa runner(void* param);
void* reindeer runner(void* param);
void* elves runner(void *param);
void int sig handler(int signum);


int main()
{
   signal(SIGINT,int sig handler);
   sem init(&mutex, 0, 1);
   sem init(&elf sem, 0, 1);
   sem init(&santa sem, 0, 0);
   sem init(&reindeer sem, 0, 0);
   int i=0,r[9];
   pthread t santa thread,elves threads[9],reindeer threads[9];

   pthread create(&santa thread,NULL,santa runner,NULL);
   for(int i=0;i<9;++i)
   {
       r[i]=i;
       pthread create(&reindeer threads[i],NULL,reindeer runner,&r[i]);
       pthread create(&elves threads[i],NULL,elves runner,&r[i]);
   }

   pthread join(santa thread,NULL);
   for(int i=0;i<9;++i)
   {
       pthread join(reindeer threads[i],NULL);
       pthread join(elves threads[i],NULL);
   }

   return 0;
}
```

```
void* santa runner(void* param)
{
   while(1)
   {
       sem wait(&santa sem);
       sem wait(&mutex);
       if(reindeer>=9)
       {
           printf("sleigh ready!! merry christmas!!\n");
           for(int i=0;i<9;++i)
               sem post(&reindeer sem);
           reindeer=0;
       }
       else if(elves==3)
           printf("santa is helping the elves\n");
       sem post(&mutex);
   }
   pthread exit(0);
}

void* reindeer runner(void* param)
{
   int* id=(int *) param;
   int reindeer threads=*id;
   printf("reindeer %d came\n",reindeer threads);
   while(1)
   {
       sem wait(&mutex);
       reindeer+=1;
       if(reindeer==9)
           sem post(&santa sem);
       sem post(&mutex);
       sem wait(&reindeer sem);
       printf("reindeer %d is getting hitched\n",reindeer threads);
       sleep(2);
   }
   pthread exit(0);
}
```

```
void* elves runner(void *param)
{
   int* id=(int *) param;
   int reindeer threads=*id;
   while(1)
   {
       sem wait(&elf sem);
       sem wait(&mutex);
       elves+=1;
       if(elves==3)
           sem post(&santa sem);
       else
           sem post(&elf sem);
       sem post(&mutex);
       printf("elf %d needs help!\n",reindeer threads);
       sleep(1);

       sem wait(&mutex);
       elves-=1;
       if(elves==0)
           sem post(&elf sem);
       sem post(&mutex);
       sleep(1);
   }
   pthread exit(0);
}


void int sig handler(int signum)
{
   printf("\nexiting...\n");
   exit(0);
}
```

output in next page...

```
santi@edith:~/OS/L8$ gcc santa-claus.c -pthread
santi@edith:~/OS/L8$ ./a.out
reindeer 0 came
elf 0 needs help!
reindeer 1 came
reindeer 2 came
elf 1 needs help!
santa is helping the elves
elf 2 needs help!
reindeer 4 came
reindeer 5 came
reindeer 3 came
reindeer 8 came
reindeer 7 came
reindeer 6 came
sleigh ready!! merry christmas!!
reindeer 0 is getting hitched
reindeer 7 is getting hitched
reindeer 1 is getting hitched
reindeer 4 is getting hitched
reindeer 3 is getting hitched
reindeer 8 is getting hitched
reindeer 5 is getting hitched
reindeer 6 is getting hitched
reindeer 2 is getting hitched
elf 3 needs help!
elf 4 needs help!
elf 5 needs help!
santa is helping the elves
elf 2 needs help!
elf 6 needs help!
elf 7 needs help!
santa is helping the elves
sleigh ready!! merry christmas!!
reindeer 0 is getting hitched
reindeer 3 is getting hitched
reindeer 1 is getting hitched
reindeer 4 is getting hitched
reindeer 6 is getting hitched
reindeer 2 is getting hitched
reindeer 5 is getting hitched
reindeer 7 is getting hitched
```

```
reindeer 8 is getting hitched
elf 0 needs help!
elf 8 needs help!
elf 1 needs help!
santa is helping the elves
elf 4 needs help!
elf 3 needs help!
elf 5 needs help!
santa is helping the elves
sleigh ready!! merry christmas!!
reindeer 0 is getting hitched
reindeer 1 is getting hitched
reindeer 4 is getting hitched
reindeer 5 is getting hitched
reindeer 2 is getting hitched
reindeer 7 is getting hitched
reindeer 6 is getting hitched
reindeer 3 is getting hitched
reindeer 8 is getting hitched
elf 2 needs help!
elf 7 needs help!
elf 6 needs help!
santa is helping the elves
elf 8 needs help!
elf 0 needs help!
elf 1 needs help!
santa is helping the elves
sleigh ready!! merry christmas!!
reindeer 0 is getting hitched
reindeer 1 is getting hitched
reindeer 3 is getting hitched
reindeer 7 is getting hitched
reindeer 5 is getting hitched
reindeer 4 is getting hitched
reindeer 6 is getting hitched
reindeer 8 is getting hitched
reindeer 2 is getting hitched
elf 6 needs help!
elf 5 needs help!
santa is helping the elves
elf 7 needs help!
elf 2 needs help!
```

```
reindeer 8 is getting hitched
elf 0 needs help!
elf 8 needs help!
elf 1 needs help!
santa is helping the elves
elf 4 needs help!
elf 3 needs help!
elf 5 needs help!
santa is helping the elves
sleigh ready!! merry christmas!!
reindeer 0 is getting hitched
reindeer 1 is getting hitched
reindeer 4 is getting hitched
reindeer 5 is getting hitched
reindeer 2 is getting hitched
reindeer 7 is getting hitched
reindeer 6 is getting hitched
reindeer 3 is getting hitched
reindeer 8 is getting hitched
elf 2 needs help!
elf 7 needs help!
elf 6 needs help!
santa is helping the elves
elf 8 needs help!
elf 0 needs help!
elf 1 needs help!
santa is helping the elves
sleigh ready!! merry christmas!!
reindeer 0 is getting hitched
reindeer 1 is getting hitched
reindeer 3 is getting hitched
reindeer 7 is getting hitched
reindeer 5 is getting hitched
reindeer 4 is getting hitched
reindeer 6 is getting hitched
reindeer 8 is getting hitched
reindeer 2 is getting hitched
elf 6 needs help!
elf 5 needs help!
santa is helping the elves
elf 7 needs help!
elf 2 needs help!
```

(2) H2O Problem

explanation :

Initially hydroq and oxyq are locked. When an oxygen thread arrives it signals hydroq twice and waits for the 2 hydrogen threads to arrive.

When oxygen thread enters, it gets the mutex and checks if there are at least 2 hydrogens waiting. If yes, it signals two of hydrogen and also itself and then bonds. If no, it releases the mutex and waits.

When hydrogen thread enters, it gets the mutex and checks if there are at least 2 hydrogens and an oxygen waiting. If yes, it signals two of hydrogen and an oxygen and then bonds. If no, it releases the mutex and waits.

After bonding, threads wait for the barrier until all three threads have bonded. Now hydrogen threads continue with the next iteration and the oxygen thread releases the mutex.

code :

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

int bc=0,h=0,o=0,btotal;
sem t sbarrier,bmutex,hydroq,oxyq,mutex;

void bond();
void *oxygen runner(void *arg);
void *hydrogen runner(void *arg);
void barrier wait();
void barrier(int n);
void int sig handler(int);

int main()
{
   signal(SIGINT,int sig handler);
   pthread_t o_thread,h_thread_1,h_thread_2;
```

```
    sem init(&mutex,0,1);
    sem init(&oxyq,0,0);
    sem init(&hydroq,0,0);
    sem init(&sbarrier,0,0);
    sem init(&bmutex,0,1);
    btotal=3;
    pthread create(&o thread,NULL,oxygen runner,NULL);
    pthread create(&h thread 1,NULL,hydrogen runner,NULL);
    pthread create(&h thread 2,NULL,hydrogen runner,NULL);
    while(1);
}

void *oxygen runner(void *arg)
{
    while(1)
    {
        sem wait(&mutex);

        o+=1;
        if(h>=2)
        {
            sem post(&hydroq);
            sem post(&hydroq);
            h-=2;
            sem post(&oxyq);
            o-=1;
        }
        else
            sem post(&mutex);

        sem wait(&oxyq);
        printf("1 oxygen atom ready\n");
        bond();
        barrier wait();
        sem post(&mutex);
    }
}
```

```
void *hydrogen runner(void *arg)
{
   while(1)
   {
       sem wait(&mutex);
       h+=1;

       if(h>=2 && o>=1)
       {
           sem post(&hydroq);
           sem post(&hydroq);
           h-=2;
           sem post(&oxyq);
           o-=1;
       }
       else
           sem post(&mutex);

       sem wait(&hydroq);
       printf("1 hydrogen atom ready\n");
       bond();
       barrier wait();
   }
}



void bond()
{
   static int i = 0;
   i++;
   if(i%3 == 0)
       printf("water molecule %d created\n",i/3);
   sleep(2);
}
```

```
void barrier wait()
{
   sem wait(&bmutex);
   bc++;
   sem post(&bmutex);
   if(bc == 3)
       sem post(&sbarrier);
   sem wait(&sbarrier);
   sem post(&sbarrier);
}


void int sig handler(int signum)
{
   printf("\nexiting...\n");
   exit(0);
}
```

output in next page...

```
santi@edith:~/OS/L8$ gcc h2o.c -pthread
santi@edith:~/OS/L8$ ./a.out
1 oxygen atom ready
1 hydrogen atom ready
1 hydrogen atom ready
water molecule 1 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 2 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 3 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 4 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 5 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 6 created
1 oxygen atom ready
1 hydrogen atom ready
1 hydrogen atom ready
water molecule 7 created
^C
exiting...
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./a.out
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 1 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 2 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 3 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 4 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 5 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 6 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 7 created
^C
exiting...
santi@edith:~/OS/L8$
```

```
santi@edith:~/OS/L8$
santi@edith:~/OS/L8$ ./a.out
1 hydrogen atom ready
1 oxygen atom ready
1 hydrogen atom ready
water molecule 1 created
1 hydrogen atom ready
1 oxygen atom ready
1 hydrogen atom ready
water molecule 2 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 3 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 4 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 5 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 6 created
1 hydrogen atom ready
1 hydrogen atom ready
1 oxygen atom ready
water molecule 7 created
^C
exiting...
santi@edith:~/OS/L8$
```