Task 1

```
Applications  Places  System
abhigaur@signals6: /ad/
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_generic
the_data points to the integer value 6
the_data now points to the character a
the_data now points to the float(by adding the offset):
1.1
2.2
3.3
the_data now points to the float(by using array syntax):
1.1
2.2
3.3
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Task 2

```
Applications  Places  System
abhigaur@signals
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create
 Hello World! from child thread 139991122577152
 Hello World! from child thread 139991112087296
 Hello World! from child thread 139991101597440
 Hello World! from child thread 139991133067008

 main() after creating the thread.  My id is 139991133075200
 Hello World! from child thread 139991091107584
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create
 Hello World! from child thread 139932811470592
 Hello World! from child thread 139932821960448
 Hello World! from child thread 139932790490880
 Hello World! from child thread 139932800980736

 main() after creating the thread.  My id is 139932821968640
 Hello World! from child thread 139932780001024
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create
 Hello World! from child thread 140372582688512

 main() after creating the thread.  My id is 140372582696704
 Hello World! from child thread 140372561708800
 Hello World! from child thread 140372572198656
 Hello World! from child thread 140372551218944
 Hello World! from child thread 140372540729088
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

As we can, the output is quite random. This is because once the threads get created, the main() function continues executing and doesn't wait for the child threads to execute.

Task 3

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a/b/abhigaur/Desktop/E
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create
 Hello World! from child thread 139856559621888
 Hello World! from child thread 139856549132032
 Hello World! from child thread 139856538642176
 Hello World! from child thread 139856528152320

 main() after creating the thread.  My id is 139856559630080
 Hello World! from child thread 139856517662464
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create
 Hello World! from child thread 140219109455616
 Hello World! from child thread 140219088475904
 Hello World! from child thread 140219119945472
 Hello World! from child thread 140219130435328
 Hello World! from child thread 140219098965760

 main() after creating the thread.  My id is 140219130443520
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Task 4

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhig
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create

 main() after creating the thread.  My id is 139889302914816
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

We notice that the program exits before the threads get printed. This is because the sleep statement makes the program sleep for 3 seconds during which main() gets executed and once the control exits main(), the threads get automatically deleted and hence we don't see any output from any threads.

Task 5

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhi
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_create
 Hello World! from child thread 140411623589632

 main() after creating the thread.  My id is 140411623597824
 Hello World! from child thread 140411581630208
 Hello World! from child thread 140411602609920
 Hello World! from child thread 140411613099776
 Hello World! from child thread 140411592120064
```

We observe that the threads do get printed since the control exits the main() after three seconds and hence all the threads get created as well as printed during this duration.

Task 6

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_join

 Hello World!  It's me, MAIN!

main() -- After creating the thread.  My id is: 140308795123456
 Hello World!  It's me, thread #140308784625408 --
 Hello World!  It's me, thread #140308763645696 --
 Hello World!  It's me, thread #140308795115264 --
 Hello World!  It's me, thread #140308753155840 --
 Hello World!  It's me, thread #140308774135552 --

After joining, Good Bye World!
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

We observe that in the absence of sleep, the threads get created randomly but once we put the sleep function, the main gets executed first and then the threads get executed after 3 seconds. Either way, the output doesn't change. The main function walks through all the threads because of the pthread_join() function.

Task 7

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abh
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param1

 Hello World!  It's me, MAIN!
In main:  creating thread 0
In main:  creating thread 1
the value of threadid before type casting: 0 !
 Hello World!  It's me, thread # 0 !
In main:  creating thread 2
the value of threadid before type casting: 1 !
 Hello World!  It's me, thread # 1 !
In main:  creating thread 3
the value of threadid before type casting: 2 !
 Hello World!  It's me, thread # 2 !
In main:  creating thread 4
the value of threadid before type casting: 3 !
 Hello World!  It's me, thread # 3 !

 It's me MAIN -- Good Bye World!
the value of threadid before type casting: 4 !
 Hello World!  It's me, thread # 4 !
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhigaur/Desktop/EC527/La
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ gcc -pthread -o test_param1 test_param1.c
test_param1.c: In function 'PrintHello':
test_param1.c:14: warning: cast from pointer to integer of different size
test_param1.c: In function 'main':
test_param1.c:33: warning: cast to pointer from integer of different size
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param1

 Hello World!  It's me, MAIN!
In main:  creating thread 0
In main:  creating thread 1
In main:  creating thread 2
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # -5 !
the value of threadid before type casting: -5 !
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # -5 !
 Hello World!  It's me, thread # -5 !
In main:  creating thread 3
In main:  creating thread 4
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # -5 !

 It's me MAIN -- Good Bye World!
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # -5 !
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param1

 Hello World!  It's me, MAIN!
In main:  creating thread 0
In main:  creating thread 1
In main:  creating thread 2
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # 18446744073709551611
In main:  creating thread 3
the value of threadid before type casting: -5 !
the value of threadid before type casting: -5 !
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # 18446744073709551611
In main:  creating thread 4
 Hello World!  It's me, thread # 18446744073709551611
 Hello World!  It's me, thread # 18446744073709551611
the value of threadid before type casting: -5 !
 Hello World!  It's me, thread # 18446744073709551611

 It's me MAIN -- Good Bye World!
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

We observe that the function does compile but with some warnings since the size of pointer and the signed char is not the same (4 bytes and 1 bytes). However, if we cast it correctly, the output is always correct which proves the statement. On the contrary, if the casting is different in main and thread function, the output is incorrect due to casting the same variable to two different types.

Task 8

On running the program multiple times, the output does change in the sense that the threads get created randomly since the main function doesn't wait for them once they get created.

Task 9

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhigaur/Desktop/E
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ gcc -pthread -o test_param2 test_param2.c
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param2

Hello World! 3  2
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
After creating the thread.  My id is 139831376623360, i = 10
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param2

Hello World! 3  2
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
Hello World! 3  10
After creating the thread.  My id is 140137497446144, i = 10
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhigaur/Desk
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ gcc -pthread -o test_param2 test_param2.c
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param2

Hello World! 2  3
Hello World! 4  3
Hello World! 5  3
Hello World! 4  3
Hello World! 10  3
Hello World! 10  3
Hello World! 10  3
Hello World! 9  3
Hello World! 10  3
After creating the thread.  My id is 139896600872704, i = 3
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhigaur/Desktop/EC52
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param2

Hello World! 2  5
Hello World! 8  5
Hello World! 10  5
Hello World! 6  5
Hello World! 6  5
Hello World! 7  5
Hello World! 10  5
After creating the thread.  My id is 139883787224832, i = 5
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ !v
vim test_param2.c
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ gcc -pthread -o test_param2 test_param2.c
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param2

Hello World! 2  9
Hello World! 10  9
Hello World! 10  9
After creating the thread.  My id is 140352893953792, i = 9
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

We observe that on changing “i”, the output doesn’t change since we are passing by value. However, when we change ‘*g’ the output does get affected in a way that the number of threads that are created keep getting less with increasing values of g. This is because g is passed by reference and this changes the actual value of the loop control variable ‘i’ due to which the loop skips the middle values(which depend on the value of g).
This is also how we can decrease the number of threads that get created.

Task 10

```
Applications  Places  System
abhigaur@signals7: /ad/eng/users/a/b/abhigaur/D
File  Edit  View  Search  Terminal  Help
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param2

Hello World! 0  1
Hello World! 1  2
Hello World! 9  10
Hello World! 5  6
Hello World! 6  7
Hello World! 4  5
Hello World! 8  9
Hello World! 3  4
Hello World! 2  3
Hello World! 7  8
After creating the thread.  My id is 139737725675264, i = 10
abhigaur@signals7:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Task 11

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a/b/abhiga
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_param3

 Hello World!  It's me, MAIN!
In main:  creating thread 0
In main:  creating thread 1
 Hello World!  It's me, thread #0! sum = 28 message = First Message
In main:  creating thread 2
In main:  creating thread 3
 Hello World!  It's me, thread #1! sum = 29 message = Second Message
In main:  creating thread 4
 Hello World!  It's me, thread #3! sum = 31 message = Fourth Message
 Hello World!  It's me, thread #2! sum = 30 message = Third Message
In main:  creating thread 5
 Hello World!  It's me, thread #4! sum = 32 message = Fifth Message
 Hello World!  It's me, thread #5! sum = 1000 message = Sixth Message
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Task 12

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_sync1

 Hello World!  It's me, MAIN!
In main: created thread 1, which is blocked -- press a char and enter
a

   I, thread #0 (sum = 28 message = First Message) am now unblocked!

After joining

GOODBYE WORLD!
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

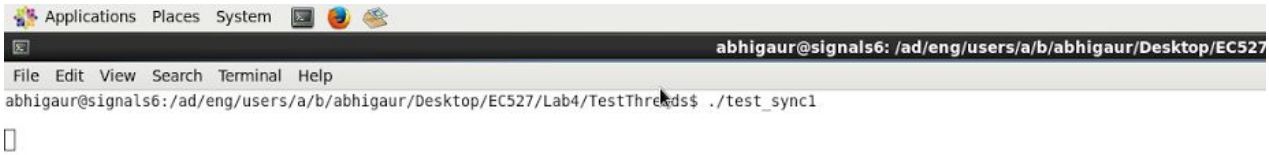
```
Applications  Places  System
abhigaur@signals6: /ad/e
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_sync1

 Hello World!  It's me, MAIN!
In main: created thread 1, which is blocked -- press a char and enter

   I, thread #0 (sum = 28 message = First Message) am now unblocked!
```

If we comment out trylock, the function doesn't wait for the the user to enter a character and the child thread gets printed straightaway since there's no lock.

Task 13



On increasing the NUM_THREADS to 2, the program doesn't work. This is because the number of threads is not equal to the number of locks. The first thread locks the execution but the second one doesn't take it since the number of locks is not equal to the number of threads.

Task 14

Applications Places System

abhigaur@signals6: /ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads

File Edit View Search Terminal Help

```
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_barrier

 Hello World!  It's me, MAIN!
In main:  creating thread 0
In main:  creating thread 1
In main:  creating thread 2
In main:  creating thread 3
In main:  creating thread 4

After creating the threads.  My id is:  140503334868736
Thread 0 printing before barrier 1 of 3
Thread 2 printing before barrier 1 of 3
Thread 1 printing before barrier 1 of 3
Thread 3 printing before barrier 1 of 3
Thread 4 printing before barrier 1 of 3
Thread 0 printing before barrier 2 of 3
Thread 4 printing before barrier 2 of 3
Thread 1 printing before barrier 2 of 3
Thread 3 printing before barrier 2 of 3
Thread 2 printing before barrier 2 of 3
Thread 0 printing before barrier 3 of 3
Thread 2 printing before barrier 3 of 3
Thread 4 printing before barrier 3 of 3
Thread 1 printing before barrier 3 of 3
Thread 3 printing before barrier 3 of 3
Thread 3 printing after barrier 1 of 2
Thread 3 printing after barrier 2 of 2
Thread 1 printing after barrier 1 of 2
Thread 1 printing after barrier 2 of 2
Thread 2 printing after barrier 1 of 2
Thread 2 printing after barrier 2 of 2
Thread 0 printing after barrier 1 of 2
Thread 4 printing after barrier 1 of 2
Thread 4 printing after barrier 2 of 2
Thread 0 printing after barrier 2 of 2

After joining
GOODBYE WORLD!
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a/b/abhigaur/Desktop/
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ gcc -pthread test_barrier.c -o test_barrier
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_barrier

 Hello World!  It's me, MAIN!
In main:  creating thread 0
In main:  creating thread 1
In main:  creating thread 2
Thread 0 printing before barrier 1 of 3
Thread 0 printing before barrier 2 of 3
Thread 0 printing before barrier 3 of 3
In main:  creating thread 3
In main:  creating thread 4

After creating the threads.  My id is:  140612857374464
Thread 1 printing before barrier 1 of 3
Thread 2 printing before barrier 1 of 3
Thread 1 printing before barrier 2 of 3
Thread 3 printing before barrier 1 of 3
Thread 1 printing before barrier 3 of 3
Thread 4 printing before barrier 1 of 3
Thread 2 printing before barrier 2 of 3
Thread 3 printing before barrier 2 of 3
Thread 2 printing before barrier 3 of 3
Thread 4 printing before barrier 2 of 3
Thread 3 printing before barrier 3 of 3
Thread 4 printing before barrier 3 of 3
Thread 4 printing after barrier 1 of 2
Thread 4 printing after barrier 2 of 2
Thread 0 printing after barrier 1 of 2
Thread 3 printing after barrier 1 of 2
Thread 3 printing after barrier 2 of 2
Thread 2 printing after barrier 1 of 2
Thread 2 printing after barrier 2 of 2
Thread 1 printing after barrier 1 of 2
Thread 1 printing after barrier 2 of 2
Thread 0 printing after barrier 2 of 2

After joining
GOODBYE WORLD!
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

We observe that the barrier works every single time but the order of the execution of statements change depending of whether we use sleep or not.

Task 15

The program works as expected. The join() gets executed after all the threads gets exited since each thread is dependent one of the previous threads and works only when the threads they are dependent on gets unlocked. So the threads get created in a specified order.

Task 16

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a/b/abhigaur/Desktop/EC527/L
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_sync2

 Hello World!  It's me, MAIN!

It's me, thread #3! I'm waiting for 1 ...
It's me, thread #4! I'm waiting for 2 ...
It's me, thread #2! I'm waiting for 1 ...
It's me, thread #6! I'm waiting 3 and 4 ...
It's me, thread #5! I'm waiting for 4 ...
It's me, thread #7! I'm waiting 5 and 6 ...
It's me, thread #8! I'm waiting 6 and 7 ...
 Created threads 2-8, type any character and <enter>
a

Waiting for thread 8 to finish, UNLOCK LOCK 1

 Done unlocking 1

It's me, thread #3! I'm unlocking 3...
It's me, thread #2! I'm unlocking 2...
It's me, thread #4! I'm unlocking 4...
It's me, thread #5! I'm unlocking 5...
It's me, thread #6! I'm unlocking 6...
It's me, thread #7! I'm unlocking 7...
It's me, thread #8! I'm unlocking 8...
After joining
GOODBYE WORLD!
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Task 17

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a/b/abhig
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_crit

 Hello World!  It's me, MAIN!
 It's me, thread #2! balance = 1000
 It's me, thread #4! balance = 1000
 It's me, thread #1! balance = 999
 It's me, thread #3! balance = 999
 It's me, thread #5! balance = 999
 It's me, thread #6! balance = 1000
 It's me, thread #7! balance = 999
 It's me, thread #8! balance = 1000
 It's me, thread #9! balance = 999
 It's me, thread #10! balance = 1000

 MAIN --> balance = 1000
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

We can change the output by either increasing the number of threads or by using the function nanosleep. The output displayed is incorrect at times since the thread creation is random and if two odd/even threads get created simultaneously, the output gets changed.

Applications Places System

abhigaur@signals6: /ad/eng/users/a/

File Edit View Search Terminal Help

```
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_crit

 Hello World!  It's me, MAIN!
 It's me, thread #1! balance = 999
 It's me, thread #2! balance = 1000
 It's me, thread #4! balance = 1001
 It's me, thread #7! balance = 999
 It's me, thread #3! balance = 998
 It's me, thread #8! balance = 1000
 It's me, thread #6! balance = 999
 It's me, thread #10! balance = 1000
 It's me, thread #5! balance = 1000
 It's me, thread #9! balance = 999

 MAIN --> balance = 1000
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Applications Places System

abhigaur@signals6: /ad/eng/users/a/b/abhigaur/Des

File Edit View Search Terminal Help

```
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_crit

 Hello World!  It's me, MAIN!
 It's me, thread #1! balance = 999
 It's me, thread #2! balance = 1000
 It's me, thread #4! balance = 1001
 It's me, thread #7! balance = 999
 It's me, thread #3! balance = 998
 It's me, thread #8! balance = 1000
 It's me, thread #6! balance = 999
 It's me, thread #10! balance = 1000
 It's me, thread #5! balance = 1000
 It's me, thread #9! balance = 999

 MAIN --> balance = 1000
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

Task 18

```
Applications  Places  System
abhigaur@signals6: /ad/eng/users/a/b/abhigaur/D
File  Edit  View  Search  Terminal  Help
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$ ./test_crit

 Hello World!  It's me, MAIN!
 It's me, thread #3! balance = 999
 It's me, thread #1! balance = 999
 It's me, thread #2! balance = 1000
 It's me, thread #7! balance = 999
 It's me, thread #5! balance = 999
 It's me, thread #6! balance = 1000
 It's me, thread #8! balance = 1000
 It's me, thread #4! balance = 1000
 It's me, thread #9! balance = 999
 It's me, thread #10! balance = 1000

 MAIN --> balance = 1000
abhigaur@signals6:/ad/eng/users/a/b/abhigaur/Desktop/EC527/Lab4/TestThreads$
```

On using mutex(), we can lock and unlock the threads as and when required and hence we get the right balance every time.