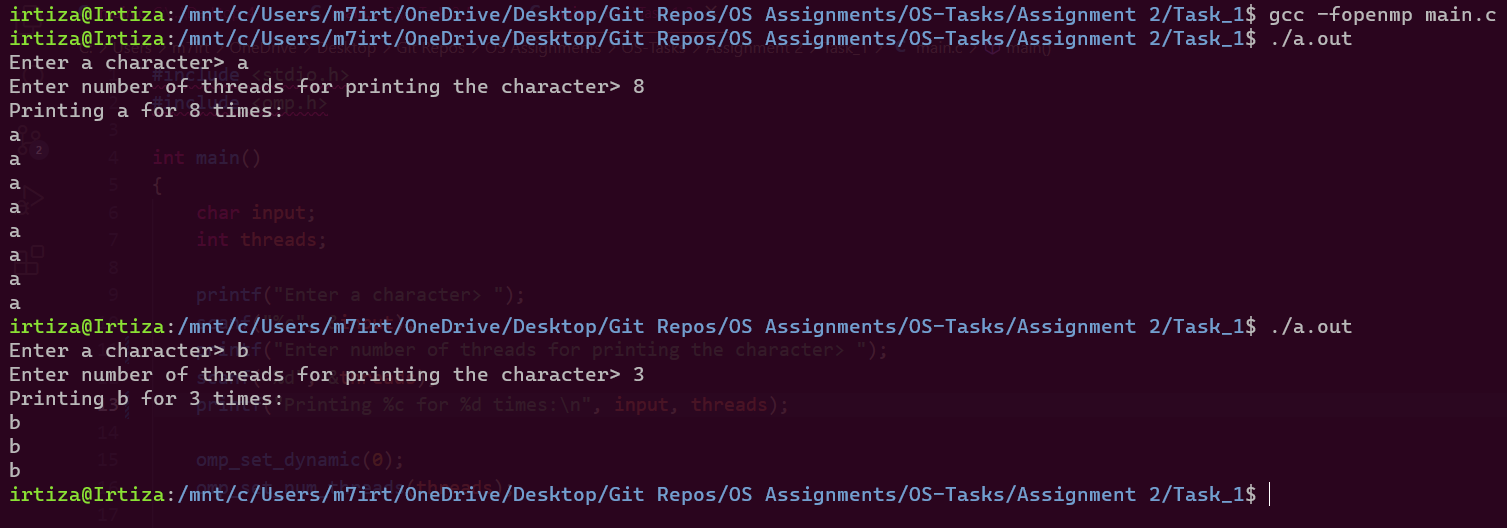
**Question # 1:**

**Method:**

The program takes a character and the number of threads from the user. I have used OpenMp to do multithreading. I set the variable “omp\_set\_dynamic” to 0 and set the variable “omp\_set\_num\_threads” to the required number of threads. In the end I added the “printf” instruction to the “parallel” construct of OpenMP.

**Output:**

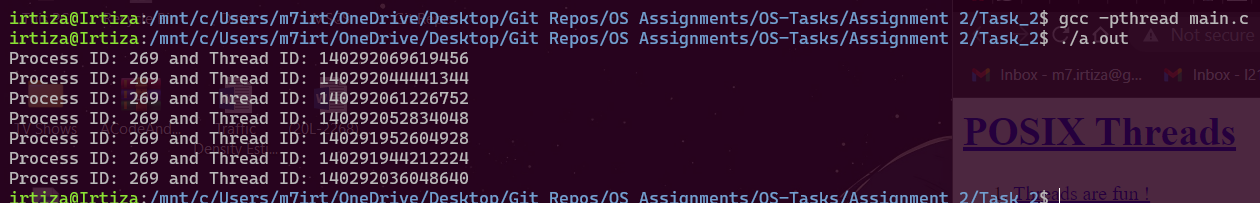
****

**Question # 2:**

**Method:**

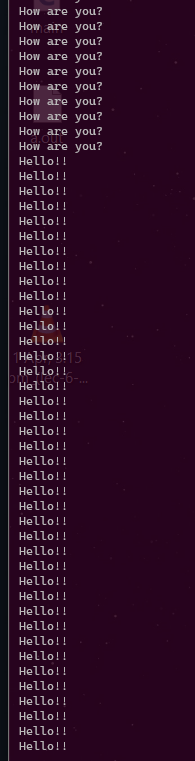
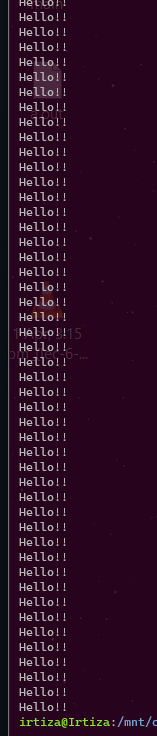
I created an array of type “pthread\_t” and size equal to 7. A for-loop is used to create 7 threads and the function passed to the thread, to be executed, prints “process id” and “thread id” using “getpid()” and “pthread\_self()” functions respectively.

**Output:**

****

**Question # 3:**

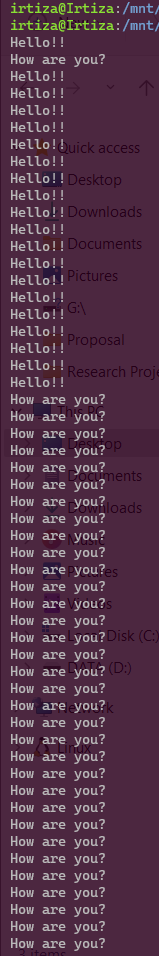
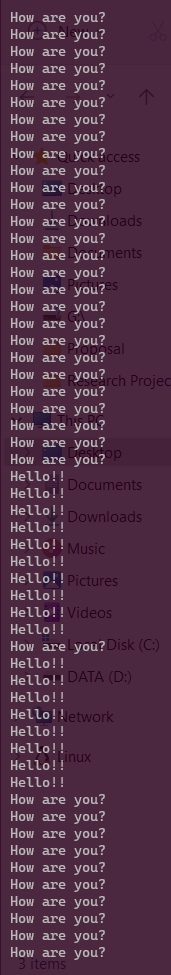
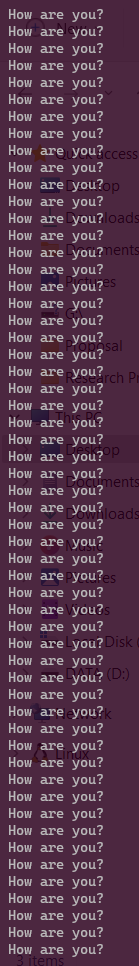
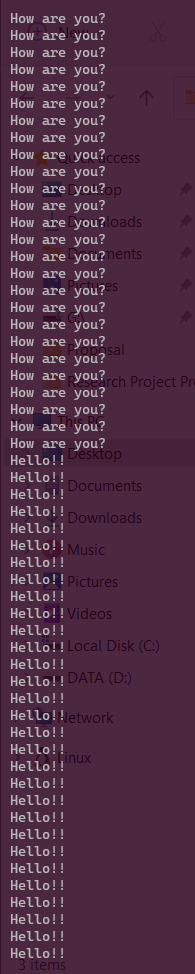
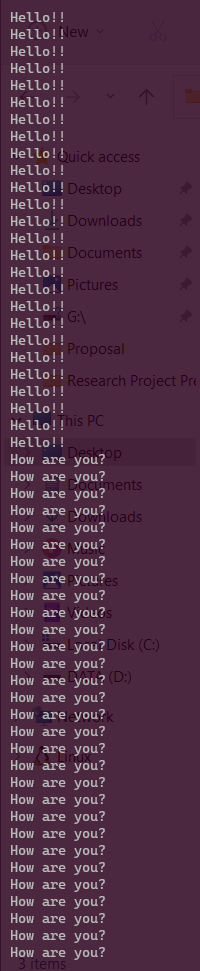
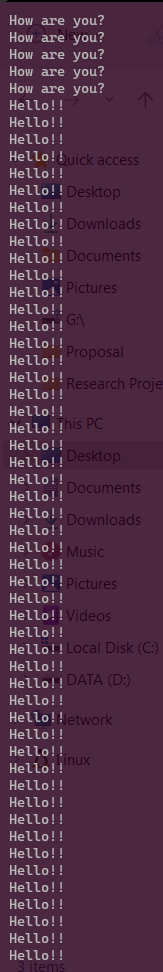
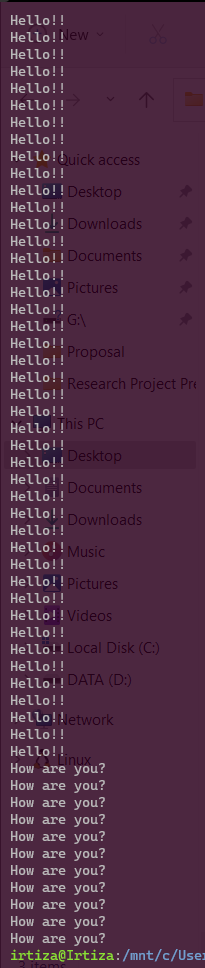
**Part A**

**Explanation:**

The output of the program is not synchronized. We can see that the **thread1()** was executed for 7 times and printed “Hello!!” but the system then scheduled the **thread2()** to run which printed “How are you?” for some number of times. Then again **thread1()** was executed in the end.

**Part B**

**      **

**Explanation:**

As the output shows the scheduling of 4 threads was not uniform. Some data from a thread was printed then it was context switched and the data from another thread was printed.

**Question # 4:**