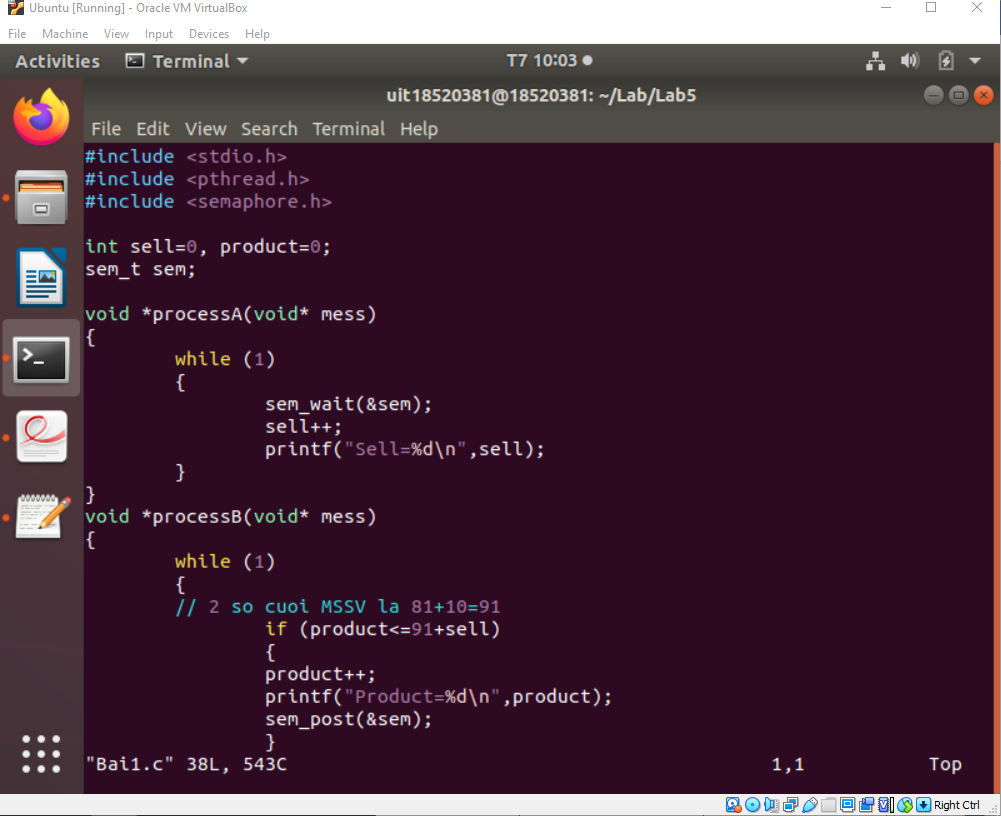
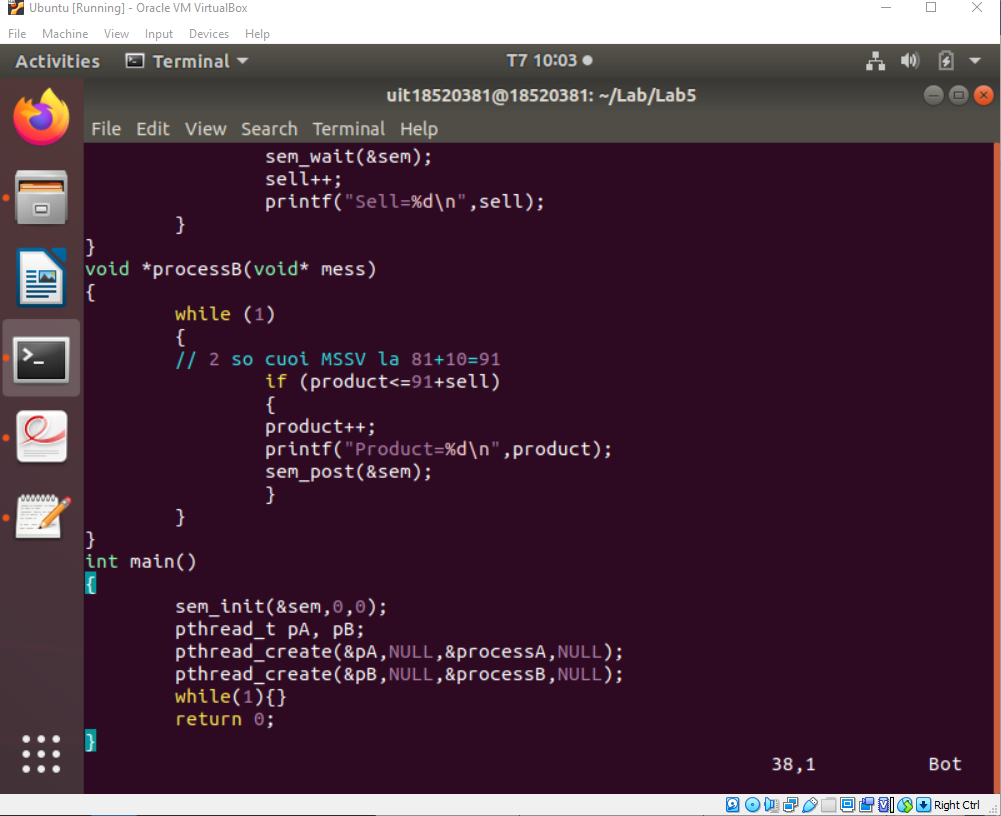
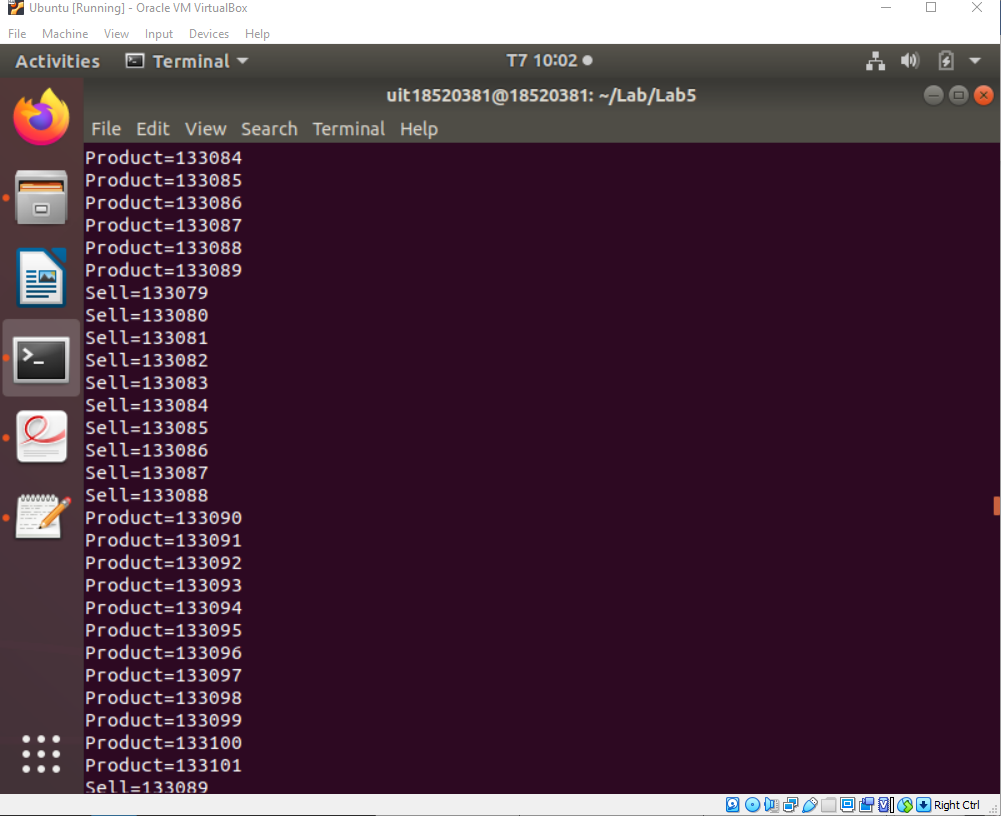
Bài 1:

Do đề bài yêu cầu sells <= products <= sells + [2 số cuối của MSSV + 10]

-> Trong process của product, trước khi tăng product lên thì ta cần phải xét xem có thỏa mãn điều kiện đề bài hay không. Các phần còn lại vẫn giữ nguyên





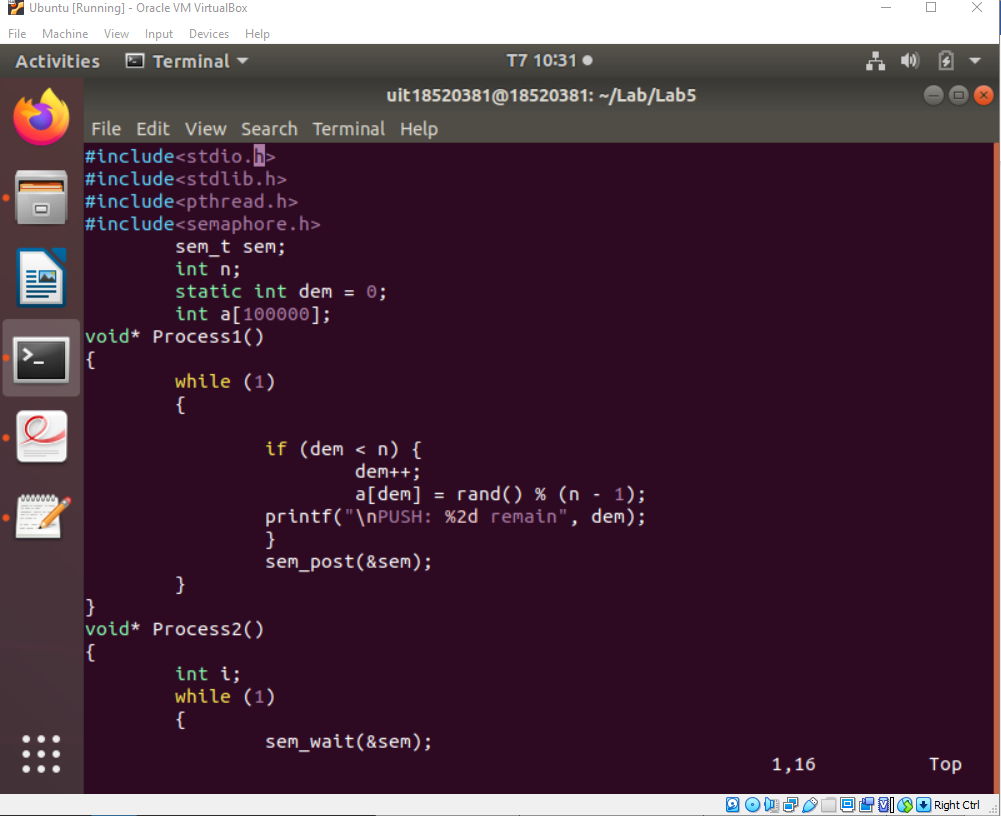


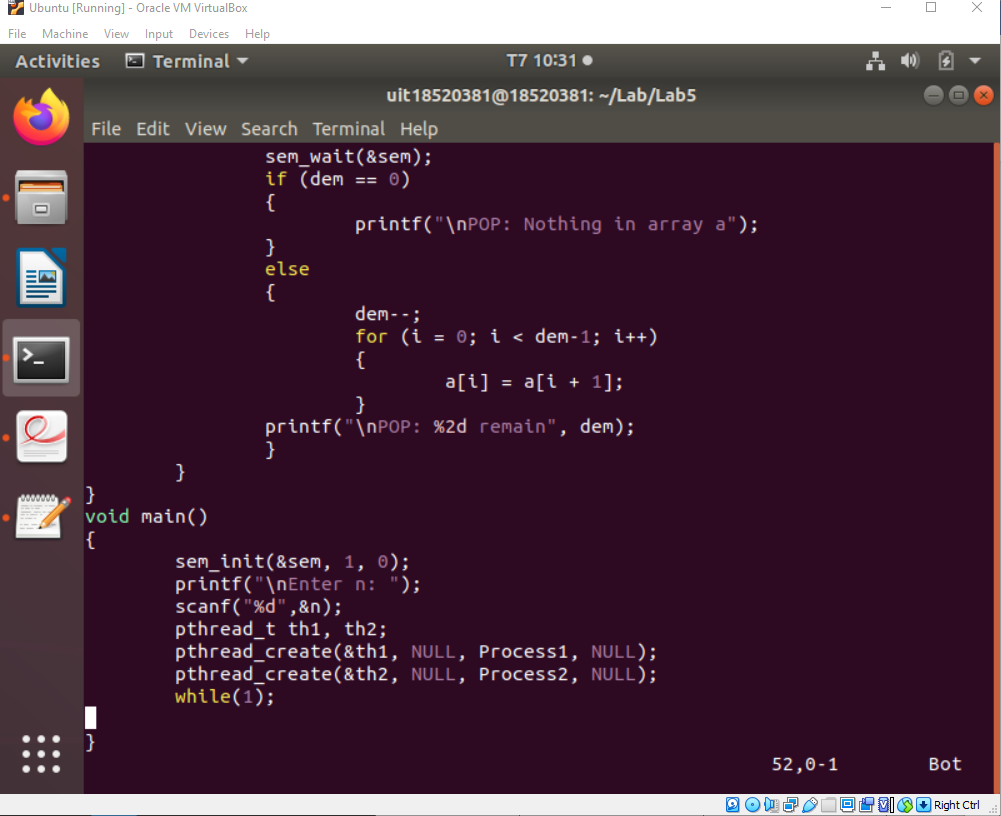
Bài 2:

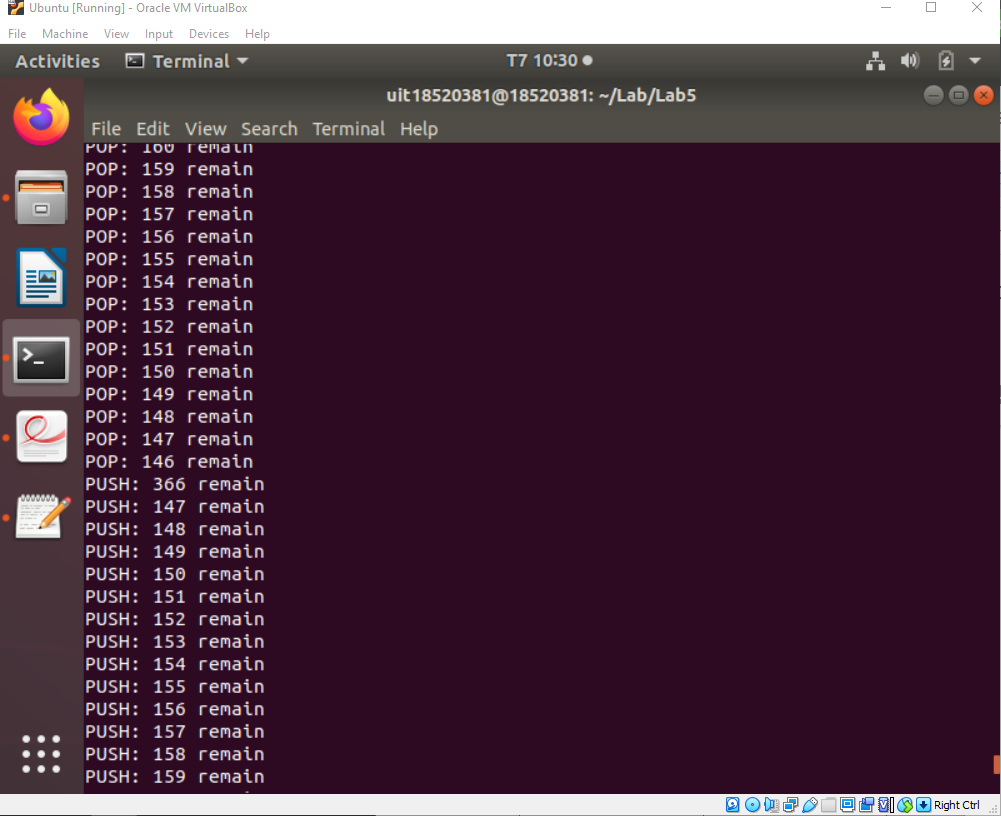
Khi thực hiện process push thì ta gán 1 phần tử mới trong mảng bằng 1 số ngẫu nhiên <n và dem++

Khi thực hiện process pop thì ta xét xem còn phần tử nào trong mảng hay không (dem>0), nếu có thì ta lấy phần tử đầu ra, dồn các phần tử phía sau lên, và dem—

Các phần còn lại cũng tương tự như 5.3.1.2



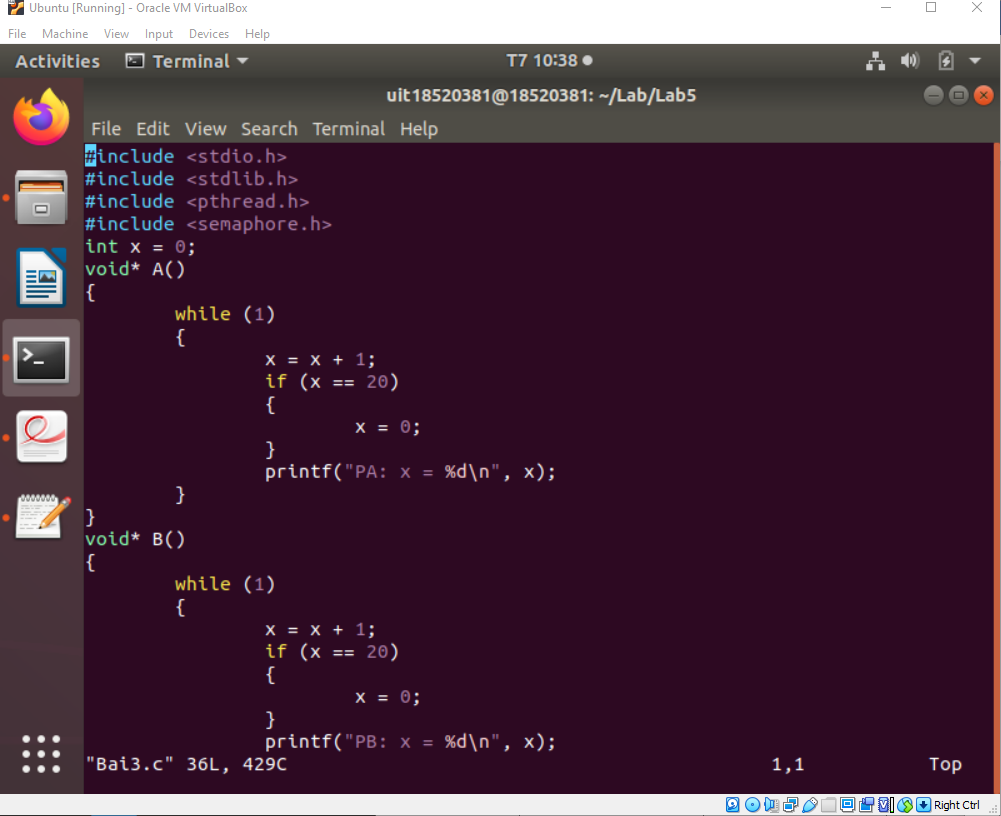


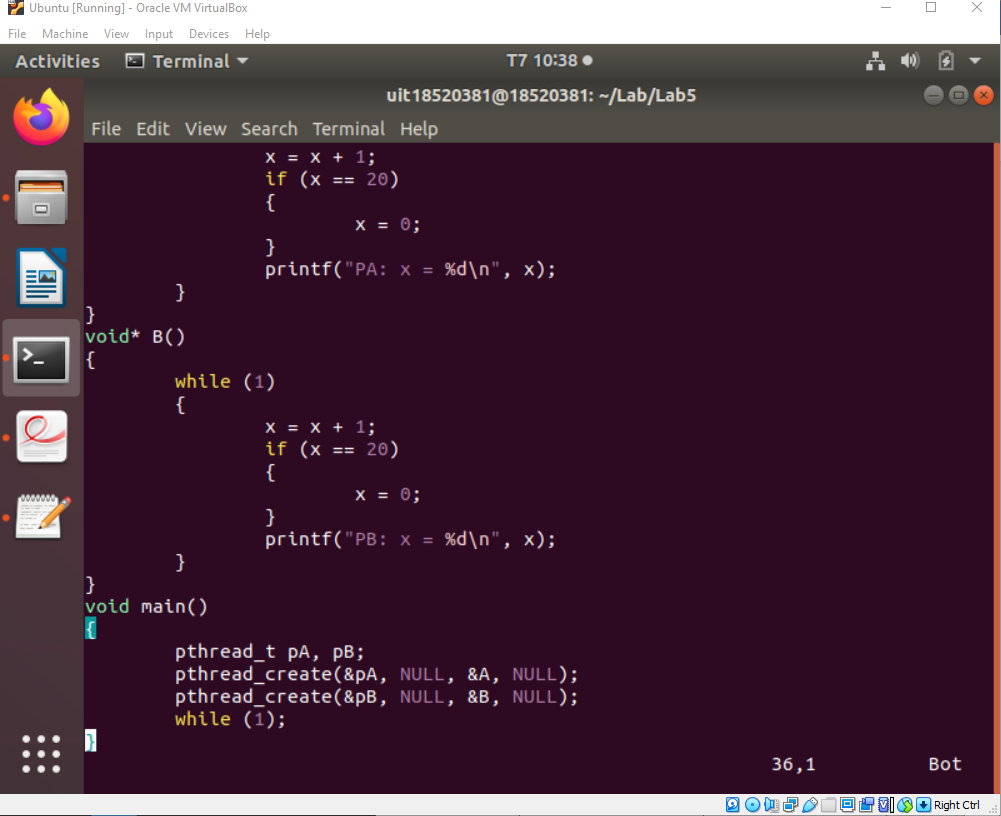


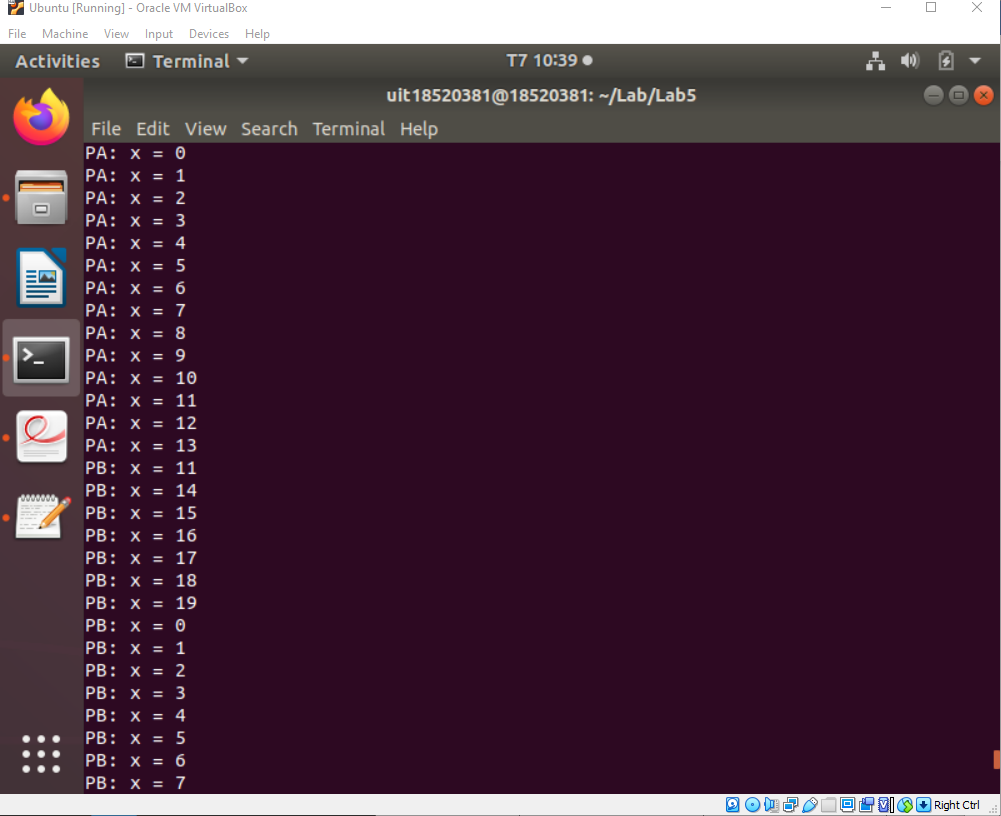
Bài 3:

Bài 3 cũng khá giống bài 2, chỉ thay dem = x và không cần mảng

Cả 2 bài 2,3 cùng chung lỗi chưa được đồng bộ hóa, và ta có xử lí lỗi này bằng mutex ở bài 4

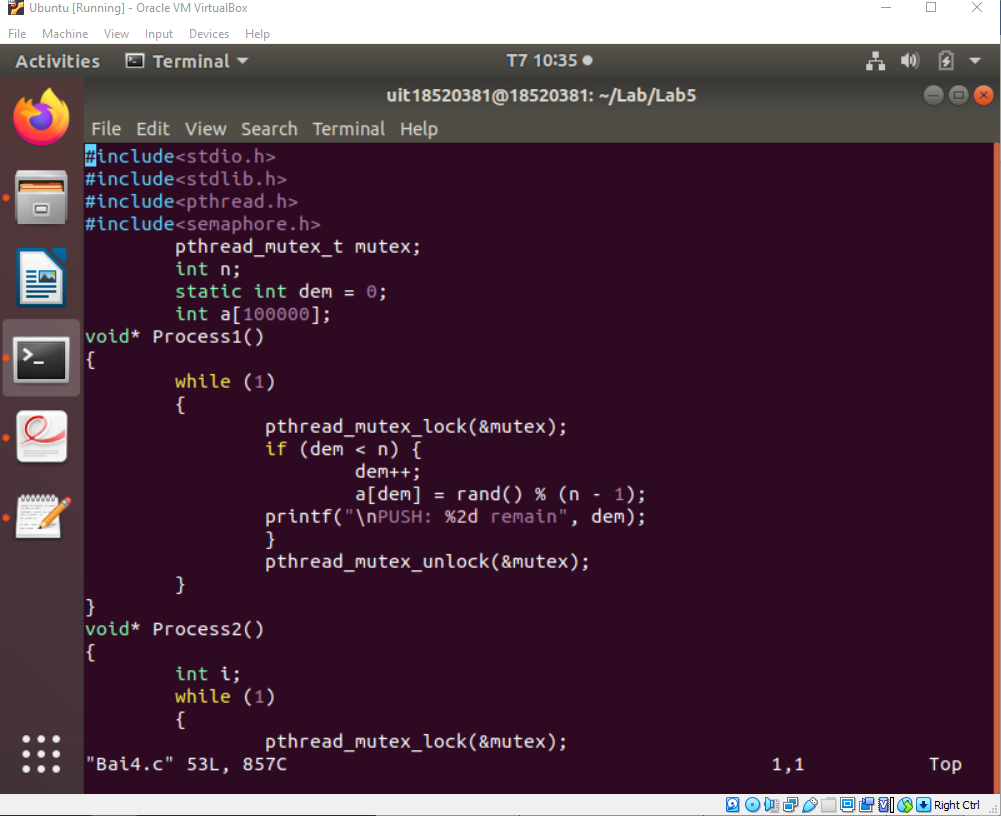


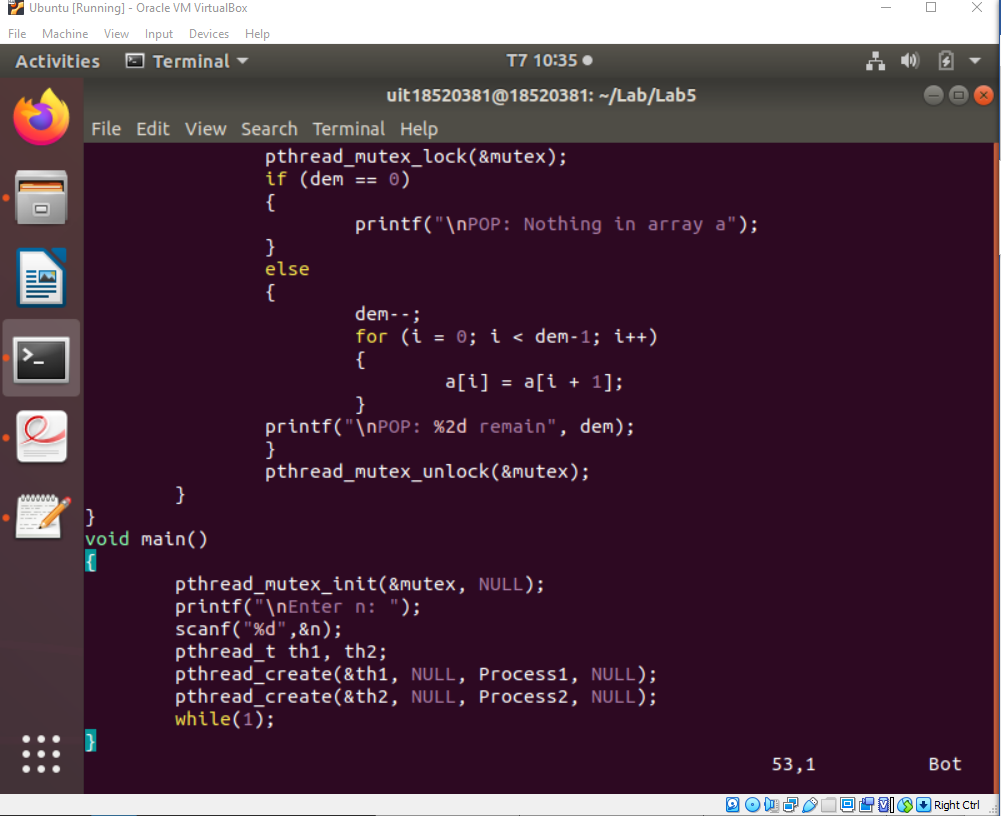


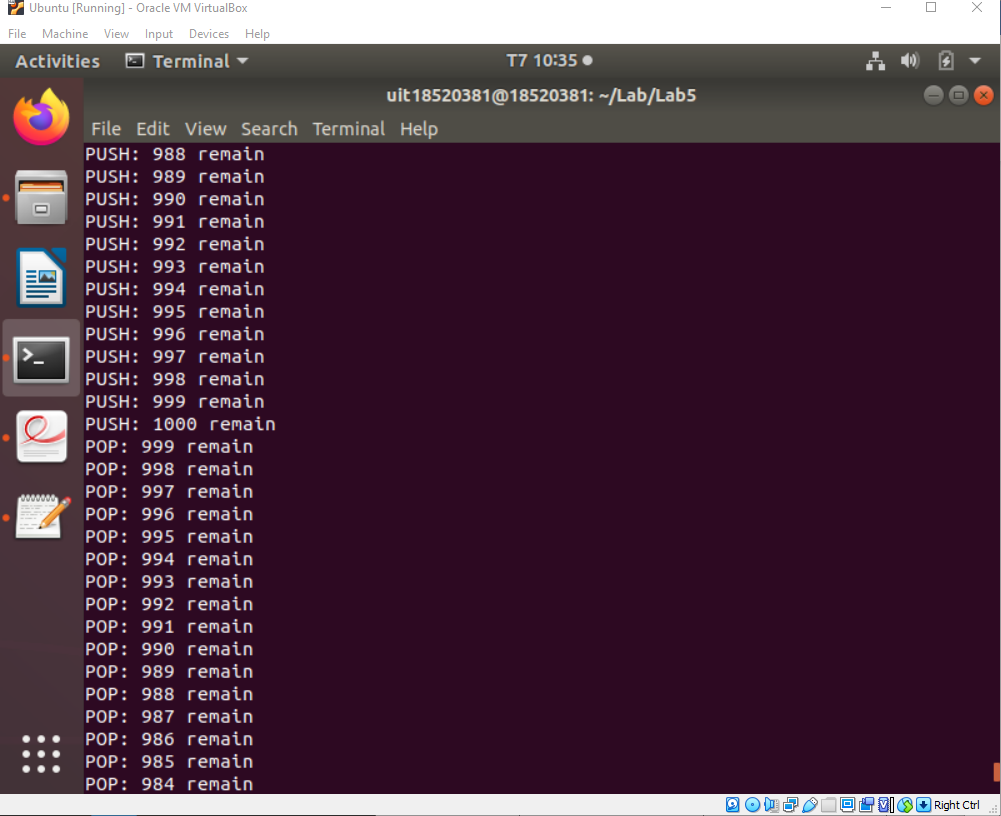


Bài 4:

Ta dùng mutex để lock process đang chạy lại, khi pop (hoặc push) được sử dụng, thì ta vẫn giữ nguyên được dem ở 1 giá trị, không còn lấy 2 giá trị như trước nữa







Bài 5:

Gọi các lệnh a,b,c,d,e,f,g là process 1,2,3,4,5,6,7

Ta có:

c), (d) chỉ được thực hiện sau khi v được tính

(e) chỉ được thực hiện sau khi w và y được tính

(g) chỉ được thực hiện sau khi y và z được tính

Thì ta dễ dàng thấy được:

-Process 3 cần process 2 để tính

-Process 4 cần process 2 để tính

-Process 5 cần process 1,3 để tính

-Process 6 cần process 1,4 để tính

-Process 7 cần process 5,6 để tính

Ta lập các semephore dùng để chặn các phép tính cho đến khi nó đủ điều kiện để thực hiện.

Khởi tạo các biến x,v,y,w,z, và các xi rồi xuất ans ở process 7

