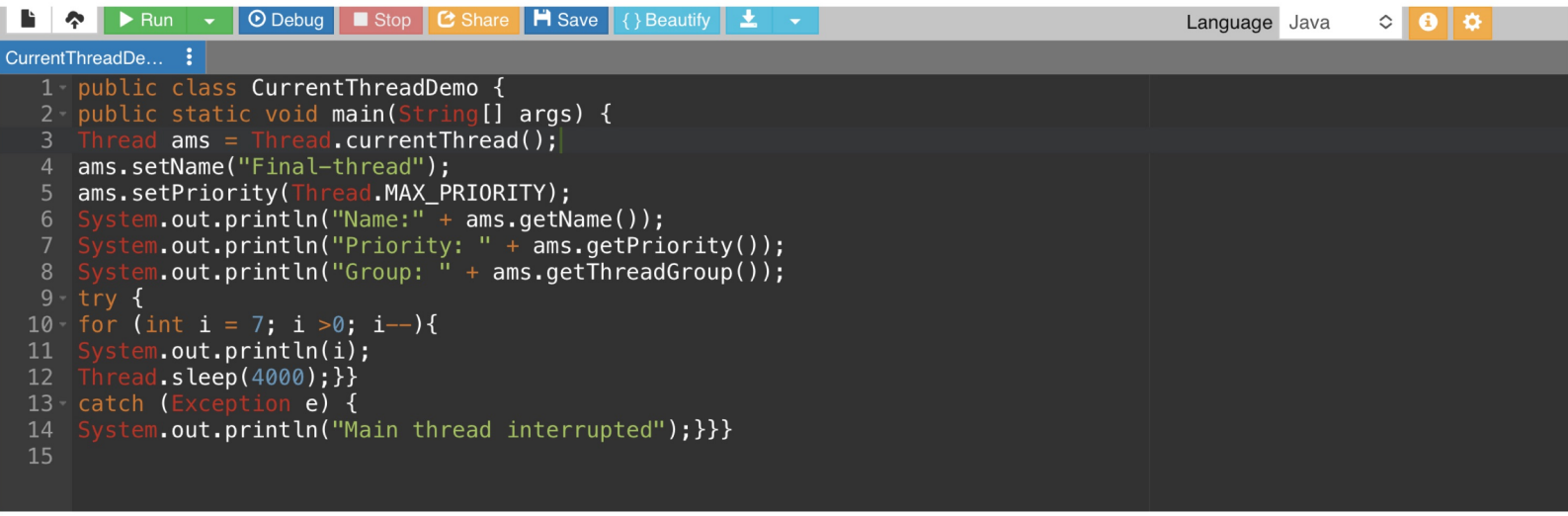
:Q1: Write a Java program achieving the tasks below

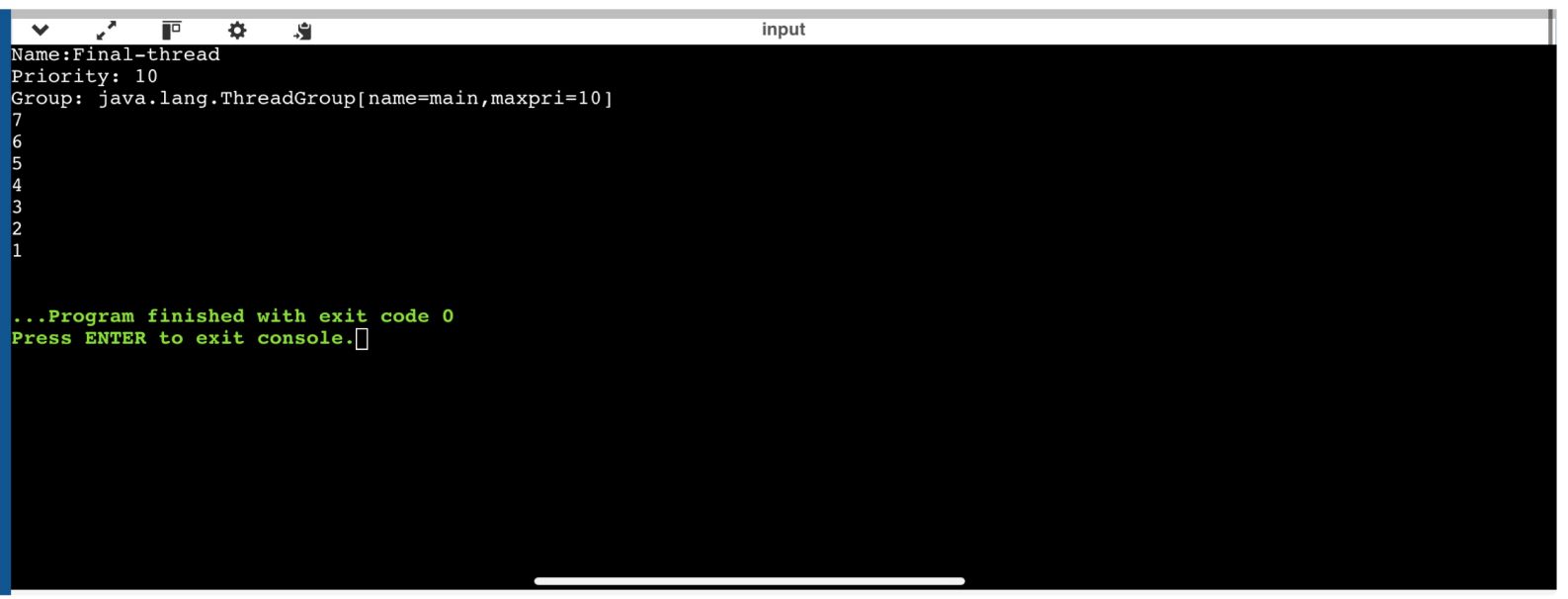
."Change the name of the main thread to "Final-thread

.Change the priority of the current thread to MAX_PRIORITY

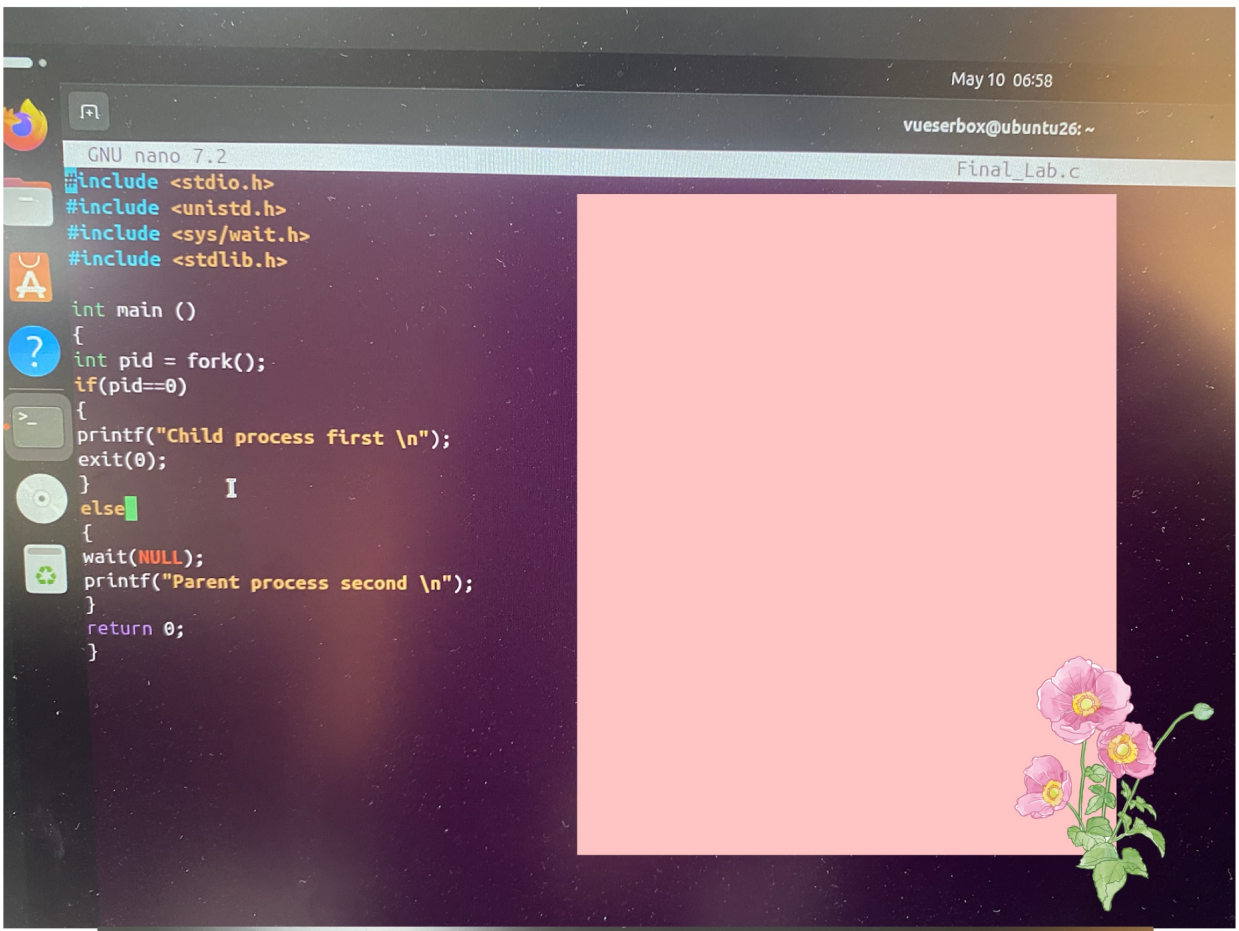
.Print the name, priority, and group of the current thread

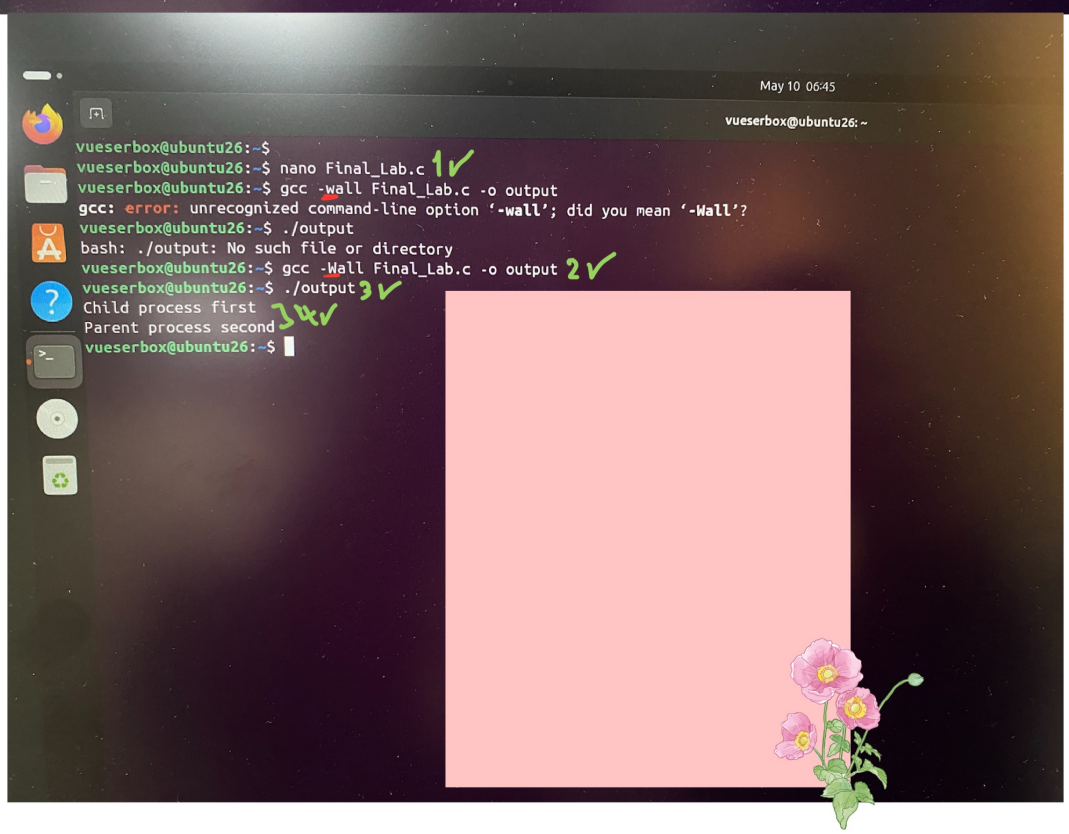
Output the numbers from 7 to 1, one number per 4 seconds (using try-catch statement)





Q2: Using the nano editor, write a C program that creates a child process such that the child prints exactly “Child process first” and then terminates immediately, while the parent waits until the child finishes execution and then prints exactly “Parent process seconed”, ensuring the child’s message is displayed before the parent’s message; name your file Final_Lab.c, save and exit, then compile and run it to verify the output .order on the terminal





Q3: Using the given Java files for the bank-account threads, fill in the missing synchronization terms in the BankAccount class only to correctly coordinate deposit and withdraw so the program avoids deadlock; the missing terms are the ones that ensure deposit and withdraw execute safely in a shared context; after completing the code, compile and run the program to observe that it runs without .deadlock

File BankAccount.java

```
public class BankAccount{
 private double balance;

  public BankAccount(){
     balance = 0;
  }

  public ..... void deposit(double amount){
     double newBalance = balance + amount;
     System.out.println("Depositing " + amount +", new balance is: " + newBalance);
     balance = newBalance;
     .....;
  }

  public ..... void withdraw(double amount){
     try{
     while (balance < amount)
     .....;
     }catch (Exception e){}
     double newBalance = balance - amount;
     System.out.println("Withdrawing " + amount+", new balance is: " + newBalance);
     balance = newBalance;
  }

  public double getBalance(){
     return balance;
  }
}
```

File WithdrawRunnable.java

```
public class WithdrawRunnable implements Runnable{
   private static final int DELAY = 200;
   private BankAccount account;
   private double amount;

   public WithdrawRunnable(BankAccount anAccount, double anAmount){
      account = anAccount;
      amount = anAmount;
   }
   public void run(){
      try{
         account.withdraw(amount);
         Thread.sleep(DELAY);
      }catch (InterruptedException exception) {}
   }
}
```

File DepositRunnable.java

```
public class DepositRunnable implements Runnable{
   private static final int DELAY = 200;
   private BankAccount account;
   private double amount;

   public DepositRunnable(BankAccount anAccount, double anAmount){
      account = anAccount;
      amount = anAmount;
   }
```

```
   public void run(){
      try{
         account.deposit(amount);
         Thread.sleep(DELAY);
      }catch (InterruptedException exception) {}
   }
}
```

File BankAccountThreadRunner.java

```
public class BankAccountThreadRunner{
   public static void main(String[] args){
      BankAccount account = new BankAccount();
      final double AMOUNT = 100;
      final int THREADS = 25;

      for (int i = 1; i <= THREADS; i++){
         DepositRunnable d = new DepositRunnable(account, AMOUNT);
         WithdrawRunnable w = new WithdrawRunnable(account, AMOUNT);
         Thread dt = new Thread(d);
         Thread wt = new Thread(w);
         dt.start();
         wt.start();
      }
   }
}
```

Run | Debug | Stop | Share | Saved | { } Beautify | Language Java

BankAccount.java | WithdrawRunnable... | DepositRunnable.j... | BankAccountThrea...

```java
public class BankAccount{
private double balance;
public BankAccount(){
balance = 0;
}
//First space
public synchronized void deposit(double amount){
double newBalance = balance + amount;
System.out.println("Depositing " + amount +", new balance is: " + newBalance);
balance = newBalance;
//Second spac
notifyAll();
}
//Third space
public synchronized void withdraw(double amount){
try{
while (balance < amount)
//Fourth space
wait();
}catch (Exception e){}
double newBalance = balance - amount;
System.out.println("Withdrawing " + amount+", new balance is: " + newBalance);
balance = newBalance;
}
public double getBalance(){
return balance;
}}

```

```
Depositing 100.0, new balance is: 100.0
Withdrawing 100.0, new balance is: 0.0
Depositing 100.0, new balance is: 100.0
Withdrawing 100.0, new balance is: 0.0
Depositing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Depositing 100.0, new balance is: 200.0
Withdrawing 100.0, new balance is: 100.0
Withdrawing 100.0, new balance is: 0.0


...Program finished with exit code 0
Press ENTER to exit console.
```