**1) Find next greater number with same set of digits**

Given a number n, find the smallest number that has same set of digits as n and is greater than n. If x is the greatest possible number with its set of digits, then print “not possible”.

Examples:

For simplicity of implementation, we have considered input number as a string.

Input: n = "218765"

Output: "251678"

Input: n = "1234"

Output: "1243"

Input: n = "4321"

Output: "Not Possible"

Input: n = "534976"

Output: "536479"

Following is the algorithm for finding the next greater number.

I) Traverse the given number from rightmost digit, keep traversing till you find a digit which is smaller than the previously traversed digit. For example, if the input number is “534976”, we stop at 4 because 4 is smaller than next digit 9. If we do not find such a digit, then output is “Not Possible”.

II) Now search the right side of above found digit ‘d’ for the smallest digit greater than ‘d’. For “534976?, the right side of 4 contains “976”. The smallest digit greater than 4 is 6.

III) Swap the above found two digits, we get 536974 in above example.

IV) Now sort all digits from position next to ‘d’ to the end of number. The number that we get after sorting is the output. For above example, we sort digits in bold 536974. We get “536479” which is the next greater number for input 534976.

2) What is Critical Section in OS?

Critical section is a piece of code that accesses a shared resource (data structure or device) that must not be concurrently accessed by more than one thread of execution. A critical section will usually terminate in fixed time, and a thread, task or process will have to wait a fixed time to enter it (aka bounded waiting). Some synchronization mechanism is required at the entry and exit of the critical section to ensure exclusive use, for example a semaphore.

***Solution to the Critical Section Problem*** must meet three conditions...

1. **mutual exclusion**: if process http://www2.cs.uregina.ca/~hamilton/courses/330/notes/synchro/img5.gif is executing in its critical section, no other process is executing in its critical section
2. **progress**: if no process is executing in its critical section and there exists some processes that wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in the decision of which will enter its critical section next, and this decision cannot be postponed indefinitely
   * if no process is in critical section, can decide quickly who enters
   * only one process can enter the critical section so in practice, others are put on the queue
3. **bounded waiting**: there must exist a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted
   * The wait is the time from when a process makes a request to enter its critical section until that request is granted
   * in practice, once a process enters its critical section, it does not get another turn until a waiting process gets a turn (managed as a queue)

* A semaphore is a value in a designated place in operating system (or [kernel](http://searchenterpriselinux.techtarget.com/definition/kernel)) storage that each process can check and then change.
* A *race condition* occurs when two threads access a shared variable at the same time. The first thread reads the variable, and the second thread reads the same value from the variable. Then the first thread and second thread perform their operations on the value, and they race to see which thread can write the value last to the shared variable. The value of the thread that writes its value last is preserved, because the thread is writing over the value that the previous thread wrote.
* A **process**, in the simplest terms, is an executing program. One or more **threads** run in the context of the **process**. A **thread** is the basic unit to which the operating system allocates processor time. A **thread** can execute any part of the **process** code, including parts currently being executed by another **thread**.