1.

```
from itertools import permutations
n=int(input())
m=int(input())
grt_ele=[]
n=list(str(n))
C=0
perm=list(permutations(n))
res = list(map("".join, perm))
for i in res:
    if int(m)<int(i):
        grt_ele.append(i)
        C=1
if C == 0:
    print -1
else:
    print(min(grt_ele))
```

3. Reference(
https://www.codespeedy.com/find-the-longest-possible-route-in-a-matrix-with-hurdles-in-python/
)

```
import sys
#Function for finding longest possible route in the matrix with hudles
#If the destination is not reachable function returns false
#Source Cell=>(i,j)
#Destination Cell =>(x,y)
def LongestPath(mat,i,j,x,y,visited):

    #Extracting Rows and columns of matrix mat
    C=len(mat[1])
    R=len(mat)
    #if source and destination are same return true
    if i==x and j==y:
        p=[True,0]
        return p

    #if cell is not valid
    if (i<0 or i>=R or j<0 or j>=C or mat[i][j]==0 or visited[i][j]):
        p=[False,sys.maxsize]
        return p
    #including (i,j) in current path
```

```
    #or setting visited(i,j) to true
    visited[i][j]=True
    #storing longest path from current cell (i,j) to destination cell (x,y)
    res=-sys.maxsize-1
    #go left from current cell
    sol=LongestPath(mat,i,j-1,x,y,visited)
    #update res => only if destination can be reached on going left from current cell
    if (sol[0]==True):
        res=max(res,sol[1])
    #go right from current cell
    sol=LongestPath(mat,i,j+1,x,y,visited)
    #update res => only if destination can be reached on going right from current cell
    if (sol[0]== True):
        res=max(res,sol[1])
    #go up from current cell
    sol=LongestPath(mat,i-1,j,x,y,visited)
    #update res => only if destination can be reached on going up from current cell
    if (sol[0]== True):
        res=max(res,sol[1])
    #go down from current cell
    sol=LongestPath(mat,i+1,j,x,y,visited)
    #update res => only if destination can be reached on going down from current cell
    if (sol[0]== True):
        res=max(res,sol[1])
    #Backtrack
    visited[i][j]= False
    #return True => if destination can be reached from current cell
    if (res != -sys.maxsize-1):
        p=[True,1+res]
        return p
    #return False => if destination can't be reached from current cell
    else :
        p=[False, sys.maxsize]
        return p
#Wrapper function
def FindLongestPath(mat,i,j,x,y):
    #Extracting Rows and columns of matrix mat
    C=len(mat[1])
    R=len(mat)
    #initializing a matrix visited that will keep a track with all Falses initially of cells visited
    visited=[[False for X in range (C)]for Y in range(R)]
    #find longest route from source to destination and printing its maximum cost
    p=LongestPath(mat,i,j,x,y,visited)
```

```
    if (p[0]):
        print("LENGTH OF LONGEST POSSIBLE ROUTE: ",p[1])
    #if destination is not reachable
    else:
        print("SORRY! DESTINATION CAN'T BE REACHED")
#Driver Code
#Input Matrix
mat=[[1,1,1,1,1,1,1,1,1,1],[1,1,0,1,1,0,1,1,0,1],[1,1,1,1,1,1,1,1,1,1]]
#Finding longest path
#Source => (0,0)
#Destination => (1,7)
FindLongestPath(mat,0,0,1,7)
```