```
# non recursive
Sour = [1, 2, 3, 4, 5, 6, 7, 8, 9]
Aux = []
Des = []
n = 9
while True:
  if n == 0:
    while len(Aux) != 0:
      Des.append(Aux.pop())
    break
  else:
    Aux.append(Sour.pop())
    n -= 1

print(Des)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
#recursive
Sour = []
Aux = []
Des = []
def TOH(n, Sour, Aux, Des):
  if n == 1:
    popped = Sour.pop()
    Des.append(popped)
    print("Mode disk "+ str(popped) + " from Source to Destination")
    while len(Aux) != 0:
      popped = Aux.pop()
      Des.append(popped)
      print("Mode disk "+ str(popped) + " from Auxilliary to Destination")

    print("Source contains the disks : " + str(Sour))
    print("Auxilliary contains the disks : " + str(Aux))
    print("Destination contains the disks : " + str(Des))
  else:
    popped = Sour.pop(0)
    print("Move disk " + str(popped) + " from Source to Auxilliary.")
    Aux.append(popped)
    TOH(n-1, Sour, Aux, Des)

n = int(input("Enter the number of disks: "))
# disk creator for Source
for i in range(1, n+1):
  Sour.append(i)
TOH(n, Sour, Aux, Des)
```

```
Enter the number of disks: 9
Move disk 1 from Source to Auxilliary.
Move disk 2 from Source to Auxilliary.
Move disk 3 from Source to Auxilliary.
Move disk 4 from Source to Auxilliary.
Move disk 5 from Source to Auxilliary.
Move disk 6 from Source to Auxilliary.
Move disk 7 from Source to Auxilliary.
Move disk 8 from Source to Auxilliary.
Mode disk 9 from Source to Destination
Mode disk 8 from Auxilliary to Destination
Mode disk 7 from Auxilliary to Destination
Mode disk 6 from Auxilliary to Destination
Mode disk 5 from Auxilliary to Destination
Mode disk 4 from Auxilliary to Destination
Mode disk 3 from Auxilliary to Destination
Mode disk 2 from Auxilliary to Destination
Mode disk 1 from Auxilliary to Destination
Source contains the disks : []
Auxilliary contains the disks : []
Destination contains the disks : [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Supplemental Activity

1. Explain the programming paradigms/techniques (like recursion or dynamic programming) that you used to solve the given problem

Answer: I only used recursion for my solution because I don't know how can I apply memoiazation or tabulation to make it more efficient. In explaining my recursive approach, the TOH function will continue to be invoked where it will transfer the all the disks from the source to auxilliary except the last one on the source stack. If there is only one disk remaining in the source, the base case will be triggered where the last disk in source will be transferred to the destination and also all of the disks in the auxilliary.

2. Provide screenshots of the techniques and provide a quick analysis.

Answer: Unfortunely I don't have data on my phone so I can't attach it to this documentation but I will send in the comments later. The explanation for the techniques is pretty much the same in number 1.