Procedure

Uses recursive steps to solve the problem under the 'solveTOH' method, taking user inputs by assigning an object to instantiate the method with the parameters defined.

```
def solveTOH(n, source, auxiliary, target):                  #n is the number of disks.
  if n == 0:                                                 #Base case if there are no disk
    print("Enter any non-zero integer!")
  elif n == 1:                                               #Base case if there is only one
    print(f"Place disk 1 from {source} to {target}")         #it on the destination peg.
  else:
    solveTOH(n-1, source, target, auxiliary)                 #Move n-1 disk from source to a
    print(f"Place disk {n} from {source} to {target}")
    solveTOH(n-1, auxiliary, source, target)                 #Move n-1 disk from auxiliary t

solve = int(input("Enter Number of Disks: "))                #Object to instantiate class wi
solveTOH(solve, 'Peg1(A)', 'Peg2(B)', 'Peg3(C)')
```

```
Enter Number of Disks: 3
Place disk 1 from Peg1(A) to Peg3(C)
Place disk 2 from Peg1(A) to Peg2(B)
Place disk 1 from Peg3(C) to Peg2(B)
Place disk 3 from Peg1(A) to Peg3(C)
Place disk 1 from Peg2(B) to Peg1(A)
Place disk 2 from Peg2(B) to Peg3(C)
Place disk 1 from Peg1(A) to Peg3(C)
```

Supplementary Activity

With the use of dynamic programming, memoization is also implemented to solve the problem with the recursive function 'solveTOH' wherein similar to the Procedure, 'n' is the number of disks with the other parameters as source, auxiliary, and target as pegs, with the 'memo' to store results of memoization.

With 'n-1' as the main recursive step, the recursion follows as the topmost block that is transferred to the destination peg, with the next being the secondmost towards the auxiliary peg, and then finally transferring the blocks towards the destination peg which then completes the problem.

```
def solveTOH(n, source, auxiliary, target, memo):
    if n == 0:
        return []
    elif n == 1:
        return [(1, source, target)] #Places the only disk from the source to target peg.

    elif (n, source, auxiliary, target) not in memo: #For all disks not in the memo yet,
        memo[(n, source, auxiliary, target)] = (solveTOH(n-1, source, target, auxiliary,
        + [(n, source, target)]
        + solveTOH(n-1, auxiliary, source, target, memo)) #Move n-1 disk from auxiliary t
    return memo[(n, source, auxiliary, target)]

def main():
    n = int(input("Enter the number of disks: ")) #n is the number of disks.
    if n == 0:  #Base case if there are no disks, prints afterward.
      print("Enter a non-zero positive integer!")
    elif n <= 0: #Base case if input is less than or equal to zero, prints afterward.
      print("Enter a positive, non-zero integer!")

    memo = {} #Memoization dictionary to store values.
    moves = solveTOH(n, 'Peg1(A)', 'Peg2(B)', 'Peg3(C)', memo)

    for disk, source, target in moves: #Assign 'disk', 'source', 'target' in moves from m
      print(f"Move disk {disk} from {source} to {target}")

if __name__ == "__main__":
    main()
```

```
Enter the number of disks: 3
Move disk 1 from Peg1(A) to Peg3(C)
Move disk 2 from Peg1(A) to Peg2(B)
Move disk 1 from Peg3(C) to Peg2(B)
Move disk 3 from Peg1(A) to Peg3(C)
Move disk 1 from Peg2(B) to Peg1(A)
Move disk 2 from Peg2(B) to Peg3(C)
Move disk 1 from Peg1(A) to Peg3(C)
```