QUESTION 1

1. (a) Write a code for the Tower of Hanoi Algorithm using two temporary pegs I.e. There is a source pole/peg named T1, two temporary pegs named T2 and T3; a destination pole/peg named T4. Take the number of disks as input from the user.

Please find the attached code file `2022129_Q1.cpp`.

1. (b) Give the sequence of moves required to move 8 disks from source to destination, using the above algorithm. Explain how the code is working with 8 disks.

The following is the sequence of moves required to move 8 disks from source to destination, using the attached algorithm in file 2022129_Q1.cpp

T1 -> T4	T1 -> T2	T4 -> T2	T1 -> T3	T1 -> T4	T3 -> T4	T2 -> T1	T2 -> T4
T1 -> T4	T1 -> T3	T1 -> T2	T3 -> T2	T4 -> T2	T4 -> T1	T2 -> T1	T4 -> T3
T4 -> T2	T3 -> T2	T1 -> T4	T1 -> T2	T4 -> T2	T1 -> T3	T1 -> T4	T3 -> T4
T2 -> T1	T2 -> T4	T1 -> T4	T2 -> T3	T2 -> T1	T3 -> T1	T4 -> T2	T4 -> T1
T2 -> T1	T2 -> T3	T2 -> T4	T3 -> T4	T1 -> T4	T1 -> T2	T4 -> T2	T1 -> T3
T1 -> T4	T3 -> T4	T2 -> T1	T2 -> T4	T1 -> T4			

1. (c) Compare the answer you have received in Q1.2 with the answer you will receive with the traditional Tower of Hanoi Setup, compute and compare their time complexities.

Using 2 temporary pegs:

The recurrence relation for the algorithm used in **1. (a)** involving 1 source, 2 temporary and 1 destination pegs is:

$$T(n) = 2\ T(n-2) + 3$$

1. `T(n-2)` : we first take the upper n-2 disks from the source (T1) and move them to T2, the first temporary peg.
2. `1` : then we move the (n-1)-th disk from the source and move it to T3, the second temporary peg.
3. `1` : then we move the (n)-th disk from the source (T1) and move it to T4, the destination peg.
4. `1` : then we move the (n-1)-th disk from T3 and move it to T4, the destination peg.
5. `T(n-2)` : then we first take the (n-2) disks from T2 and move them to the T4, the destination peg.

Using this method it took us *45 ($(3 * 2^{8/2}) - 3$) steps to move 8 disks from T1 to T4*.

Time Complexity: $\theta(2^{n/2})$

$T(n) \Rightarrow 2\ T(n-2) + 3$

$\Rightarrow 2^2\ T(n-4) + 6 + 3$

$\Rightarrow 2^3\ T(n-6) + 3[2^2 + 2^1 + 1]$

$\Rightarrow \ldots$

$\Rightarrow 2^k T(n - 2k) + 3[2^{k-1} + \ldots + 2^2 + 2^1 + 1]$ (assume $T_2=3$ & $(n-2k) = 2$)
$\Rightarrow 2^{(n/2)-1}(3) + 3\,[2^{(n/2)-2} + \ldots + 2^2 + 2^1 + 2^0]$
$\Rightarrow 3\ [2^{(n/2)-1} + \ldots + 2^2 + 2^1 + 2^0]$
$\Rightarrow 3\,(2^{n/2} - 1)$

Therefore, for n levels time complexity = $\theta(3*2^{n/2} - 3) \approx \theta(2^{n/2})$

Using 1 temporary peg:

The recurrence relation for the traditional Tower of Hanoi algorithm involving 1 source, 1 temporary and 1 destination peg is:

$$T(n) = 2\ T(n-1) + 1$$

1. `T(n-1)` : we first take the upper n-1 disks from the source (T1) and move them to T2, the temporary peg.
2. `1` : then we move the (n)-th disk from the source (T1) and move it to T3, the destination peg.
3. `T(n-1)` : then we first take the (n-1) disks from T2 and move them to the T3, the destination peg.

Using this method it took us *255 (2^8-1) steps to move 8 disks from T1 to T3*.

Time Complexity: $\theta(2^n)$

T(n) $\Rightarrow 2\ T(n-1) + 1$
$\Rightarrow 2^2\ T(n-2) + 2 + 1$
$\Rightarrow 2^3\ T(n-3) + 2^2 + 2^1 + 1$
$\Rightarrow \ldots$
$\Rightarrow 2^k T(n - k) + [2^{k-1} + \ldots + 2^2 + 2^1 + 1]$ (assuming $T_1=1$ & $(n-k)=1$)
$\Rightarrow 2^{n-1} + 2^{n-2} + \ldots + 2^2 + 2^1 + 1$
$\Rightarrow 2^n - 1$

Therefore, for n levels time complexity = $\theta(2^n-1) \approx \theta(2^n)$

QUESTION 2

2. (a) Write a recursive and an iterative code for Merge sort. Take the array to be sorted as input from the user. Assume it to be an integer array only. Print the sorted array from both recursive call and iterative call.

Please find the attached code file `2022129_Q2.cpp`.

2. (b) Explain the iterative code by an example.

Taking the example: {32, 42, 1, 4, 32, 15, 6}

Initialization:

The given array: {32, 42, 1, 4, 32, 15, 6}.
The size of the array is 7.

First Pass:

- The outer loop of the iterative merge sort starts with pass = 1.
- In the inner loop, var = 0 and pass = 1.
- So, low = 0, high = min(1, 6) = 1, and mid = min(0 + 1 - 1, 6) = 0.
- The merge function is called with low = 0, mid = 0, and high = 1, which sorts the elements {32, 42}.
- The array becomes {32, 42, 1, 4, 32, 15, 6}.

Second Pass:

- The outer loop continues with pass = 2.
- In the inner loop, var = 0 and pass = 2.
- So, low = 0, high = min(3, 6) = 3, and mid = min(0 + 2 - 1, 6) = 1.
- The merge function is called with low = 0, mid = 1, and high = 3, which sorts the elements {1, 4, 32}.
- The array becomes {1, 4, 32, 42, 32, 15, 6}.

Third Pass:

- The outer loop continues with pass = 4.
- In the inner loop, var = 0 and pass = 4.
- So, low = 0, high = min(7, 6) = 6, and mid = min(0 + 4 - 1, 6) = 3.
- The merge function is called with low = 0, mid = 3, and high = 6, which sorts the elements {1, 4, 6, 15, 32, 32}.
- The array becomes {1, 4, 6, 15, 32, 32, 42}.

Final Merge:

- After the loop completes, there's a final merge call with low = 0, mid = (7-1)/2 = 3, and high = 6, which merges the entire array.

Sorted Array:

- The sorted array is {1, 4, 6, 15, 32, 32, 42}.

QUESTION 3

3. Arrange the following functions in a sequence f1, f2, . . . , f24 so that fi = O (f_i+1) for 1 ≤ i ≤ 23.