THE UTOPIAN TREE

The Utopian problem is a common problem where a tree grows in cycles of two phases each year.

1. SPRING (DOUBLE HEIGHT): The tree doubles it height

2. SUMMER (INCR. BY 1m): The tree grows by 1m.

The Growth pattern starts with a tree of height 1m. Given a number of cycles, you need to calculate the final height of the tree.

- **EX:-**

If no. of cycle = n

- cycle 0: Initial height = 1
- cycle 1: (Spring): Height doubles → 1 × 2 = 2
- cycle 2: (Summer): Height increases by 1 → 2 + 1 = 3
- cycle 3 (Spring): Height doubles → 3 × 2 = 6
- cycle 4 (Summer): Height increases by 1 → 6 + 1 = 7

FINAL HEIGHT AFTER 4 cycles = 7.

C++

```
int utopianTree(int n) {
    int height = 1; // initial height of the tree
    for (int i=1; i<=n; i++) {
        if (i%2==1) {
            height = height * 2; // Spring: double the height.
        } else {
            height = height+1;
        }
    }
    return height;
}
int main(){
    int cycles;
    printf("Enter the no.of cycles:");
    scanf("%d",&cycle);
    int result = utopianTree(cycles);
    printf("height of utopionTree : ", cycles, result);

    return 0;
}
```

TIME COMPLEXITY.

T.C - Iterative - O(n)
S.C - O(1)

optimized Approach :- (for Specific case) →

- odd cycles: Height is $2^K - 1$, where K is no.of complete odd cycles.
- Even cycles : Height can be derived as $2^{K+1} - 1$

FORMULA: O(1)