Alpha - Beta Pruning

```
import math

# Alpha-Beta Pruning Algorithm
def alpha_beta_search(depth, index, is_max, values, alpha, beta,
target_depth):
    """Recursive function for Alpha-Beta Pruning."""
    # Base case: If the target depth is reached, return the leaf node
value
    if depth == target_depth:
        return values[index]

    if is_max:
        # Maximizer's turn
        best = -math.inf
        for i in range(2):
            val = alpha_beta_search(depth + 1, index * 2 + i, False,
values, alpha, beta, target_depth)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break  # Prune remaining branches
        return best
    else:
        # Minimizer's turn
        best = math.inf
        for i in range(2):
            val = alpha_beta_search(depth + 1, index * 2 + i, True,
values, alpha, beta, target_depth)
            best = min(best, val)
            beta = min(beta, best)
            if beta <= alpha:
                break  # Prune remaining branches
        return best

def main():
    # User Input: Values of leaf nodes
    print("Enter the values of leaf nodes separated by spaces:")
    values = list(map(int, input().split()))
```

```
    # Calculate depth of the game tree
    target_depth = math.log2(len(values))
    if target_depth != int(target_depth):
        print("Error: The number of leaf nodes must be a power of 2.")
        return
    target_depth = int(target_depth)

    # Run Alpha-Beta Pruning
    result = alpha_beta_search(0, 0, True, values, -math.inf, math.inf,
target_depth)

    # Display the result
    print(f"The optimal value determined by Alpha-Beta Pruning is:
{result}")

if __name__ == "__main__":
    main()
    name = "Varsha Prasanth"
    usn = "1BM22CS321"
    print(f"Name: {name}, USN: {usn}")
```

OUTPUT

```
Enter the values of leaf nodes separated by spaces:
10 9 14 18 5 4 50 3
The optimal value determined by Alpha-Beta Pruning is: 10
Name: Varsha Prasanth, USN: 1BM22CS321
```