1. Write a program to implement Alpha Beta pruning in Python. The algorithm can be applied to any depth of tree by not only pruning the tree leaves but also the entire subtree. Order the nodes in the tree such that the best nodes are checked first from the shallowest node. USN-1NT18CS133

Algorithm: Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta. Alpha is the best value that the maximizer currently can guarantee at that level or above. Beta is the best value that the minimizer currently can guarantee at that level or above.

```
function minimax(node, depth, isMaximizingPlayer, alpha, beta):
  if node is a leaf node :
          return value of the node

  if isMaximizingPlayer :
          bestVal = -INFINITY
          for each child node :
                  value = minimax(node, depth+1, false, alpha, beta)
                  bestVal = max( bestVal, value)
                  alpha = max( alpha, bestVal)
                  if beta <= alpha:
                          break
          return bestVal
  else :
        bestVal = +INFINITY
        for each child node :
                value = minimax(node, depth+1, true, alpha, beta)
                bestVal = min( bestVal, value)
                 beta = min( beta, bestVal)
                 if beta <= alpha:
                          break
        return bestVal
```

In [1]:
```
import math
MIN,MAX= -1000,1000

def MINMAX(depth,nodeIndex,maximizingPlayer,values,alpha,beta):
    if depth==math.ceil(math.log(len(values),2)):
        return values[nodeIndex]
    if maximizingPlayer:
        best=MIN
        for i in range(0,math.ceil(math.log(len(values),2))-1):
            val = MINMAX(depth+1,nodeIndex*2+i,False,values,alpha,beta)
            best=max(best,val)
            alpha=max(alpha,best)
            if beta<=alpha:
                    break
        return best
    else:
        best=MAX
        for i in range(0,math.ceil(math.log(len(values),2))-1):
            val = MINMAX(depth+1,nodeIndex*2+i,True,values,alpha,beta)
            best=min(best,val)
            alpha=min(alpha,best)
            if beta<=alpha:
                    break
        return best


values=[3,4,2,9,12,5,23,23]
print("Optimal value:",MINMAX(0,0,True,values,MIN,MAX))
```

```
Optimal value: 12
```

In [ ]: