Assignment 3 : Alpha Beta Tree

```
// A simple C++ program to find
// maximum score that
// maximizing player can get.
#include<bits/stdc++.h>
using namespace std;
// Returns the optimal value a maximizer can obtain.
// depth is current depth in game tree.
// nodeIndex is index of current node in scores[].
// isMax is true if current move is
// of maximizer, else false
// scores[] stores leaves of Game tree.
// h is maximum height of Game tree
int minimax(int depth, int nodeIndex, bool isMax,
 int scores[], int h)
{
 // Terminating condition. i.e
 // leaf node is reached
 if (depth == h)
 return scores[nodeIndex];
 // If current move is maximizer,
 // find the maximum attainable
 // value

 if (isMax)
 return max(minimax(depth+1, nodeIndex*2, false, scores, h),
 minimax(depth+1, nodeIndex*2 + 1, false, scores, h));

 // Else (If current move is Minimizer), find the minimum
 // attainable value
 else
 return min(minimax(depth+1, nodeIndex*2, true, scores, h),
 minimax(depth+1, nodeIndex*2 + 1, true, scores, h));
}
// A utility function to find Log n in base 2
int log2(int n)
{
 return (n==1)? 0 : 1 + log2(n/2);
}
// Driver code
int main()
{
 // The number of elements in scores must be
```

```
 // a power of 2.
 int scores[] = {3, 5, 2, 9, 12, 5, 23, 23};
 int n = sizeof(scores)/sizeof(scores[0]);
 int h = log2(n);
 int res = minimax(0, 0, true, scores, h);
 cout << "The optimal value is : " << res << endl;
 return 0;
}
```