```fsharp
module Poolean.HuffmanDomain

open Poolean.Nucleotide.Domain
open CompressionDomain

module HuffmanDomain =

    type ASCII = char
    type Shape<'T> = | Leaf of ASCII | SubTree of 'T | Empty
    type Tree<'T> = {Frequency: int; Node: Shape<'T>}
    type Quad = {e1: Tree<Quad> option; e2: Tree<Quad> option; e3: Tree<Quad> option; e4: Tree<Quad> option}

    let quadHuffman (acc:Heap<Quad>) =
        let removeOption (x:Tree<'T> option) = match x with | Some s -> {Frequency = s.Frequency; Node = SubTree s} | None -> {Frequency = 0; Node = Empty}
        let rec buildTree (acc:Heap<Quad>) =
            match acc.Size() with
            | 1 -> acc // base case
            | _ ->
                let t1 = acc.ExtractMin()
                let t2 = acc.ExtractMin()
                let t3 = acc.ExtractMin()
                let t4 = acc.ExtractMin()

                let t = {
                    Frequency =
                        [t1; t2; t3; t4]
                        |> List.map (fun x -> match x with | Some e -> e.Frequency | None -> 0)
                        |> List.sum;
                    Node =
                        let collapse = [t1; t2; t3; t4] |> List.filter (fun x -> match x with |Some s -> s.Node <> Empty | None -> false)
                        match List.length collapse with
                        | 0 -> Empty
                        | 1 -> match collapse.Head with | Some s -> s.Node | None -> Empty
                        | _ -> SubTree {e1 = t1; e2 = t2; e3 = t3; e4 = t4}
                        }
                do acc.Insert(t)
                buildTree acc

    let rec quadTreePrinter (tree:Tree<Quad>) (prefix:Nucleotide) acc =
        match tree.Node with
        | Leaf a ->  printfn "%A\t\t%d\t\t\t%s" a tree.Frequency acc
        | SubTree t ->
            match t.e1 with
                | Some x -> quadTreePrinter x prefix.Increment (acc + (prefix.Increment).String)
                | None -> ()
            match t.e2 with
                | Some x -> quadTreePrinter x prefix.Increment.Increment (acc + prefix.Increment.Increment.String)
                | None -> ()
            match t.e3 with
                | Some x -> quadTreePrinter x prefix.Increment.Increment.Increment (acc + prefix.Increment.Increment.Increment.String)
                | None -> ()
            match t.e4 with
                | Some x -> quadTreePrinter x prefix.Increment.Increment.Increment.Increment (acc + prefix.Increment.Increment.Increment.Increment.String)
                | None -> ()
        | Empty -> ()
```