Pensieve header: The Free Lie / Lazy Evaluation Abstraction Challenge.

```
MakeLieSeries[ser_Symbol, expr_] := (
   ser[] = Hold[MakeLieSeries[ser, expr]];
   ser[d_Integer] := ser[d] = Expand[expr /. w_LW /; Deg[w] ≠ d → 0];
   LieSeries[ser]
  );

AddLieSeries[ss___LieSeries] := AddLieSeries[ss] = Module[{ser},
    ser = Unique[AddLieSeries];
    ser[] = Hold[AddLieSeries[ss]];
    ser[d_Integer] := ser[d] = Plus @@ ((#[d]) & /@ {ss});
    LieSeries[ser]
   ];

b[s1_LieSeries, s2_LieSeries] := b[s1, s2] = Module[{ser},
    ser = Unique[b];
    ser[] = Hold[b[s1, s2]];
    ser[d_Integer] := ser[d] = Sum[
       b[s1[k], s2[d - k]],
       {k, 1, d - 1}
      ];
    LieSeries[ser]
   ];

LieDerivation[der_Symbol, rules_List] := (
   der[] = Hold[LieDerivation[der, rules]];
   (der[w_LW] /; Deg[w] == 1) :=
    (der[w] = MakeLieSeries[w /. Append[rules, _LW → 0]]);
   der[w_LW] := der[w] = Module[{x, y},
      {x, y} = LyndonFactorization[w];
      AddLieSeries[b[der[x], y], b[x, der[y]]]
     ];
   der[s_LieSeries] := der[s] = Module[{ser},
      ser = Unique[LieDerivationOnLieSeries];
      ser[] = Hold[der[s]];
      ser[d_] := ser[d] = Sum[
         der[s[k]][d],
         {k, 1, d}
        ];
      LieSeries[ser]
     ];
   der[as_ASeries] := Omitted;
   der[cws_CWSeries] := Omitted;
   der[expr_][d_] :=
    Expand[expr /. {w_LW :> der[w][d], s_LieSeries :> der[s][d]}];
   LieDerivation[der]
  );
```

```
BCHBase = Module[{bch},
   bch = Unique["BCHBase"];
   bch[] = Hold[BCHBase];
   bch[1] = ⟨"x"⟩ + ⟨"y"⟩;
   bch[d_Integer] := bch[d] = Expand[Plus[
        adSeries[E^(-ad), MakeLieSeries[⟨"y"⟩]][MakeLieSeries[⟨"x"⟩]][d],
        -adSeries[(1 - E^(-ad)) / ad - 1, LieSeries[bch]][
           EulerE[LieSeries[bch]]][d]
       ] / d];
   LieSeries[bch]
  ];

JA[-1, ___] = MakeCWSeries[0];
JA[n_, y_LW, μ_LieSeries, ss_] := JA[n, y, μ, ss] = Module[
    {s, sμ, μs},
    sμ = ScaleLieSeries[s, μ];
    μs = StableApply[LieMorphism[{y → Ad[ScaleLieSeries[1, sμ]][LW[z]]}], μ];
    μs = μs // LieMorphism[{LW[z] → y}];
    IntegrateCWSeries[
     AddCWSeries[
      JA[n - 1, y, μ, s] // LieDerivation[{y → b[μs, y]}],
      div[y, μs]
     ],
     {s, 0, ss}
    ]
   ];
JA[y_LW, μ_LieSeries] := JA[y, μ] = Module[{cws, s},
    cws = Unique[JA];
    cws[] = Hold[JA[y, μ]];
    cws[d_Integer] := cws[d] = JA[d - 1, y, μ, s][d] /. s → 1;
    CWSeries[cws]
   ];
```