Descriptions of functions in DetGB Software Package

1. Module DetIdeal
   1. IsPosetIdeal
      - Calling Sequence: IsPosetIdeal(L, m, n)
      - Parameters: L - a list of minors in X which are represented by their row and column indices; m, n - the size of matrix X.
      - Description: The IsPosetIdeal command returns true if L forms a poset ideal, otherwise it returns false.

2. DiagOrder
   - Calling Sequence: DiagOrder("Scantype", m, n)
   - Parameters: "Scantype" - NWE, NWS, SEW, and SEN; m, n - the size of matrix X.
   - Description: The DiagOrder returns a lexicographic term order on $K[X]$ which are diagonal induced by the corresponding scanning variable orders, where X is an $m \times n$ matrix with $x_{ij}$ as its entries.
   - NWE: Assign the North-West corner variable $x_{11}$ of X as the greatest and assign the next greatest variable by scanning row by row to the East. Others are similarly defined.

3. AntiDiagOrder
   - Calling Sequence: AntiDiagOrder("Scantype", m, n)
   - Parameters: "Scantype" - NEW, NES, SWE, and SWN; m, n - the size of matrix X.
   - Description: The AntiDiagOrder returns a lexicographic term order on $K[X]$ which are antidiagonal induced by the corresponding scanning variable orders, where X is an $m \times n$ matrix with $x_{ij}$ as its entries.
   - NEW: Assign the North-East corner variable $x_{1n}$ of X as the greatest and assign the next greatest variable by scanning row by row to the West. Others are similarly defined.

4. DetGen
   - Calling Sequence: DetGen(m, n, t)
   - Parameters: m, n - the size of matrix X; t - a positive integer
   - Description: The DetGen command constructs the generating minors of $I_t$ of X whose entries are $x_{ij}$.

5. NormalDetGen
   - Calling Sequence: NormalDetGen(A, m, n)
   - Parameters: A - a list $[[R, r], [C, s]]$, where R, C are sequences of row and column indices respectively; r, s are sequences of nonnegative integers; m, n - matrix size.
   - Description: The NormalDetGen command constructs generators for the normal determinantal ideal. That is, the $(r_i+1)$-minors of the first $R_i$ rows and $(s_j+1)$-minors of the first $C_j$ columns.

6. CorresDetRingDelta
   - Calling Sequence: CorresDetRingDelta(A, m, n)
   - Parameters: A - a list $[[R, r], [C, s]]$ where R, C are sequences of row and column indices respectively; r, s are sequences of nonnegative integers; m, n - matrix size.
   - Description: The CorresDetRingDelta command returns $\delta$ such that the determinantal ring $B[X]/I = R(X; \delta)$ where I is the normal determinantal ideal determined by $A, m, n$.

7. AlphaT
   - Calling Sequence: AlphaT(r, t)
   - Parameters: r - a non-increasing sequence $[r_1, r_2, \cdots, r_u]$; t - an integer.
Description: The AlphaT\((r, t)\) command calculates \(\alpha_t(r) = \sum_{i \leq t} r_i\) for the given sequence \(r\).

(8) GammaT

Calling Sequence: GammaT\((r, t)\)

Parameters: \(r\) - a non-increasing sequence \([r_1, r_2, \cdots, r_u]\); \(t\) - an integer.

Description: The GammaT\((r, t)\) command calculates \(\gamma_t([r_1, r_2, \cdots, r_u]) = \sum_{i=1}^{u} \max(r_i - t + 1, 0)\) for the given sequence \(r\).

(9) DualShape

Calling Sequence: DualShape\((Seq)\)

Parameters: \(Seq\) : Shape of a standard Young tableau.

Description: The DualShape\((Seq)\) command calculates the dual shape\(\)\([Bruns and Conca, 2022]\) of a standard Young table with the given shape \(Seq\).

(10) SpecialDec

Calling Sequence: SpecialDec\((M, Len, Output\_Format)\)

Parameters:

i. \(M\) : A non-increasing sequence, set \(I_M = I_{M_1} I_{M_2} \cdots\)

ii. \(Len\) : An integer representing the degree of the terms in Gröbner basis of \(I_M\)

iii. \(Output\_Format\) : Boolean Variable: \(False\) means to return all pairs of increasing decompositions, while \(true\) means to only return pairs of special increasing decompositions. The default value is \(true\).

Description: The SpecialDec\((M, Len, Output\_Format)\) command returns pairs of special increasing decompositions corresponding to the different shapes of the standard Young tableaux.

(11) IsComparable

Calling Sequence: IsComparable\((\sigma, \tau)\)

Parameters: \(\sigma, \tau\) - The non-increasing sequences.

Description: The IsComparable\((\sigma, \tau)\) command returns whether \(\sigma\) and \(\tau\) can be comparable. Here, \(\sigma \geq \tau\) means \(\alpha_k(\sigma) \geq \alpha_k(\tau)\) for all \(k\). If \(\sigma \geq \tau\), return 1; if \(\sigma < \tau\), return 0; if \(\sigma\) and \(\tau\) are not comparable, return "Incomparable".

(12) TermIncDecs

Calling Sequence: TermIncDecs\((r)\)

Parameters: \(r\) - The \(r\)-sequence of a term.

Description: The TermIncDecs\((r)\) command calculates the result of the special increasing decomposition of \(r\).

(13) TermIncDecsLen

Calling Sequence: TermIncDecsLen\((r)\)

Parameters: \(r\) - The bottom row sequence of a term.

Description: The TermIncDecsLen\((r)\) command calculates the length of the special increasing decomposition of \(r\).

(14) GammaHat

Calling Sequence: GammaHat\((r, t)\)

Parameters: \(r\) - The bottom sequence of a term.
– Description: The GammaHat($r$, $t$) command calculates

$$\hat{\gamma}_t(r) = \{\gamma_t(\lambda) : r \text{ has an indecomposition of shape } \lambda\}$$

for the given sequence $r$.

(15) **ConstructSpcTerm**
– Calling Sequence: `ConstructSpcTerm(b)`
– Parameters: $b$ - An integer, corresponding to the ideal $I_b + 2I_b$
– Description: The `ConstructSpcTerm(b)` command constructs a bottom row sequence corresponding to a term in in $(I_b + 2I_b)$ of degree $2b + 3$.

2. **Module Ladder**
(1) **ConstructLadder**
– Calling Sequence: `ConstructLadder(U, L)`
– Parameters: $U$ - a list composed of upper corners $[c, d]$; $L$ - a list composed of lower corners $[a, b]$.
– Description: If $U$ and $L$ form a ladder, then the `ConstructLadder` command returns the input $U, L$, otherwise it returns False.

(2) **LadderGen**
– Calling Sequence: `LadderGen(L, r)`
– Parameters: $L$ - the sequence of lower corners; $r$ - a sequence of nonnegative integers.
– Description: The `LadderGen` constructs the generators of the one-sided ladder determinantal ideal $I(L, r)$.

(3) **OneLadder2Perm**
– Calling Sequence: `OneLadder2Perm(L, r)`
– Parameters: $L$ - a sequence the lower corners of a ladder, $r$ - a sequence of nonnegative integers with the same size as $L$.
– Description: The `OneLadder2Perm` command returns a vexillary permutation $w$ such that the Schubert determinantal ideal $I_w$ equals to the one-sided ladder determinantal ideal $I(L, r)$.

(4) **MixLadderGen**
– Calling Sequence: `MixLadderGen(U, L, r)`
– Parameters: $U$ - a list of upper corners; $L$ - a list of lower corners; $r$ - a nonnegative integer.
– Description: The `MixLadderGen(U, L, r)` command returns the geneators of the Mixed ladder determinantal ideal determined by $U, L, r$.

(5) **BlockGen_ind**
– Calling Sequence: `BlockGen_ind(B, T)`
– Parameters: $B$ - a sequence of $[U, L]$ where $U$ and $L$ are the sequence of upper and lower corners respectively; $r$ - a sequence of nonnegative integers.
– Description: The `BlockGen_ind` function constructs the generators of the blockwise determinantal ideal where each generator is represented by row and column indices.

(6) **BlockGen**
– Calling Sequence: `BlockGen(B, T)`
– Parameters: $B$ - a sequence of $[U, L]$ where $U$ and $L$ are the sequence of upper and lower corners respectively; $r$ - a sequence of nonnegative integers.
– Description: The `BlockGen` function constructs the generators of the blockwise determinantal ideal.
(7) **DrawLadder**
   - Calling Sequence: `DrawLadder(U, L)`
   - Parameters: `U` - a list of upper corners; `L` - a list of lower corners.
   - Description: The `DrawLadder` command draws the ladder determined by `U` and `L`.

(8) **DrawBlock**
   - Calling Sequence: `DrawBlock(B)`
   - Parameters: `B` - a sequence of `[U, L]` where `U` and `L` are the sequence of upper and lower corners respectively.
   - Description: The `DrawBlock` command draws the Blocks determined by `B`.

3. **Module Schubert**

   (1) **PermMat**
       - Calling Sequence: `PermMat(p)`
       - Parameters: `p` - a permutation.
       - Description: The `PermMat` function transforms a permutation `p` to a permutation matrix.

   (2) **EssSet**
       - Calling Sequence: `EssSet(p)`
       - Parameters: `p` - a permutation.
       - Description: The `EssSet` function computes the essential set of a permutation `p`.

   (3) **RotheDiagram**
       - Calling Sequence: `RotheDiagram(p)`
       - Parameters: `p` - a permutation.
       - Description: The `RotheDiagram` command constructs the Rothe diagram of a permutation `p`.

   (4) **DrawRothe**
       - Calling Sequence: `DrawRothe(p)`
       - Parameters: `p` - a permutation.
       - Description: The `DrawRothe` command draws the Rothe diagram of a permutation `p`.

   (5) **FultonGen**
       - Calling Sequence: `FultonGen(p)`
       - Parameters: `p` - a permutation.
       - Description: The `FultonGen` function constructs Fulton’s generators `G` for the Schubert determinantal ideal of a permutation `p`.

   (6) **ElusiveGen**
       - Calling Sequence: `ElusiveGen(p)`
       - Parameters: `p` - a permutation.
       - Description: The `ElusiveGen` function constructs elusive minors `E` for the Schubert determinantal ideal of a permutation `p`.

   (7) **RedGBSchubert**
       - Calling Sequence: `RedGBSchubert(p, ord)`
       - Parameters: `p` - a permutation; `ord` - an anti-diagonal term order.
       - Description: The `RedGBSchubert` function constructs the reduced Groebner basis for the Schubert determinantal ideal of a permutation `p` w.r.t. the anti-diagonal term order `ord`.

   (8) **KLGen**
       - Calling Sequence: `KLGen(v, w)`
       - Parameters: `v, w` - two permutations with the same size.
Description: The KLGen function constructs the generators for the Kazhdan-Lusztig ideal of two permutations $v, w$.

4. Module Combin

(1) Deletion
- Calling Sequence: `Deletion(A, p)`
- Parameters: $A$ - a standard tableau of shape $(s_1, s_2, \ldots)$; $p$ - an index such that $s_p > s_{p+1}$.
- Description: The `Deletion` function constructs a standard tableua and a number $x$.

(2) Insertion
- Calling Sequence: `Insertion(A, x)`
- Parameters: $A$ - a standard tableau of shape $(s_1, s_2, \ldots)$; $x$ - an integer.
- Description: The `Insertion` function constructs a standard tableua and an index $p$.

(3) RSK
- Calling Sequence: `RSK(T)`
- Parameters: $T$ - a non-empty standard bitableau.
- Description: The RSK function returns a two-row array $A$ and a monomial which corresponding to $A$.

(4) RSKInv
- Calling Sequence: `RSKInv(A)`
- Parameters: $A$ - a two-row array which equals to RSK($B$) for some bitableau $B$.
- Description: The RSKInv function returns a standard bitableau of a two-row array $A$.

(5) DrawBitab
- Calling Sequence: `DrawBitab(T)`
- Parameters: $T$ - a non-empty standard bitableau.
- Description: The `DrawBitab` command draws the bitableau of '$T$'.

(6) Mitosis
- Calling Sequence: `Mitosis(p)`
- Parameters: $p$ - a permutation.
- Description: The `Mitosis` command computes all the reduced pipe dreams of a permutation $p$.

(7) DrawPipeDream
- Calling Sequence: `DrawPipeDream(p)`
- Parameters: $p$ - a permutation.
- Description: The `DrawPipeDream` command draws all the reduced pipe dreams of a permutation $p$.

(8) StandardRep
- Calling Sequence: `StandardRep(X, Y, m, n)`
- Parameters: $X, Y$ - two lists with the same length; $m, n$: matrix size.
- Description: The `StandardRep` command returns the standard representation of $X \cdot Y$. 