When algebra systems perform computations they need a mechanism for managing the storage of intermediate results. If we want to compute \( F^2 + G^2 \) on two objects \( F \) and \( G \) (e.g. polynomials) there must be a place for storing the result of computing \( F^2 \) and a place for storing the result of computing \( G^2 \). These are intermediate results computed in the process of obtaining the final result. They cannot be stored in the same place (so there must be at least two temporary locations for this computation). After the two intermediate results are added, the main computation is finished. The storage used for the intermediate results is no longer needed for the computation and can be made available to the system for other computations.

In our first attempt at a polynomial package, we had the operations \( P+ \), \( P- \), and \( P* \) put their results in a single box called RESULT whose address we put on the stack. This works fine if all we plan to do is perform one operation and print the result. It will not work for chain computations. In class we discussed several mechanisms for dealing with the problem: (1) explicit manipulation of temporaries by name (this includes explicit allocation and deallocation); (2) a ring buffer; (3) a polynomial stack; (4) a storage pool with garbage collection.

The name "ring buffer" suggest the image of a collection of polynomial-size slots (we are using PSize as the size of a polynomial in bytes) arranged in a circle.

Each time we need a temporary storage location we get the address of the current slot (at the pointer) and the pointer is advanced to the next slot. The pointer continues to advance around the ring. If we allow for sufficiently many slots and we write our program so that the slots are only used for temporary storage, the hope is that we will not come around the ring and allocate a slot that contains important information. [The ring buffer is not a mini-warehouse! It is only for short-term storage of intermediate results. Information of long-term interest must be promptly moved out of the ring.]

SKETCH: A "ring" is realized by a block of memory in the dictionary (use ALLOT). You need the address, say Ring.Bot ( -- addr ) of the start of the block and Ring.Top ( -- addr ) to give the address of the end of the block. Allow for 30 slots of size Psize (the size of a polynomial). You need a pointer (a VARIABLE or VALUE), say Ring.Ptr ( -- addr ) that gives the address of the current slot. Each time a temporary is called for, let PTemp return the address given by Ring.Ptr and then advance Ring.Ptr. If it advances past Ring.Top reset it to Ring.Bot.

The file POLYS.SEQ contains a polynomial package with a storage pool/garbage collection mechanism. Delete this storage mechanism and replace it with your ring buffer. You can actually chose your own names and details for the internals of the ring buffer. The only thing necessary for compatibility with the later parts of POLYS.SEQ is that you must have a word PTemp ( -- addr ) which returns the address of the next available storage location and initializes it to the zero polynomial. You can compare results using your altered POLYS.SEQ with results obtained using the original version to test if your ring buffer seems to work properly.

IF YOU CHOOSE TO DO THIS PROJECT AND NEED FURTHER INFORMATION ABOUT THE RING BUFFER OR ABOUT POLYS.SEQ, PLEASE ASK!!!