Chapter 4 Homework

Due date: Nov. 30, 2017

Program Exercises

1. Design and build a linked allocation system to represent and manipulate polynomials. You should use circularly linked lists with header nodes. Each term of the polynomial is represented as a node of the following structure:

coef	expon	link

In order to erase polynomials efficiently, use the available space list and associated functions discussed in the courses.

Write a function, *pread* to read in a polynomial and convert it to its circular representation, and return a pointer to the header node of this polynomial. In addition, write a function, *pwrite*, to output the polynomial pointed by a certain pointer (to the header node) using the following form.

	1st term (largest expon of nonzero coef)	…	Last term (smallest expon of nonzero coef)
coef		…	
expon		…	

Requirements:

a. For the function *pread*, declare a circular linked list whose header node is pointed by a pointer, and initially the pointer points to the available space list (*avail*) is pointed to NULL. A new space for a node can be created only if *avail* points to NULL.
b. In *pread,* allow users to input the coef and expon of each term to build a polynomial.
c. The function *pwrite* should output all nodes not in the available list space.

2. In **Program Exercise 1**, write a function *pmult* to compute *c=a*b*, where *a* is a pointer points to a polynomial represented using a circular linked list with a header node (*a* points to the header node), *b* is a pointer points to another polynomial represented using a circular linked list with a header node (*b* points to the header node).

Requirements:

a. Use *pread* to construct your *a* and *b*.
b. *c* is a pointer points to the header node of a linked list storing the result. Initially, *c* only contains a header node and the *avail* pointer of *c* points to NULL.

c. Using *pwrite* to output *a*, *b*, and *c*.

3. In **Program Exercise 2**, write a function *eval* to evaluate *c* at a given *x*, which is a floating point constant.
   Requirements:
   a. Let *c* (in **Program Exercise 2**) and *x* be the inputs of *eval*.
   b. Output the result of the evaluation in the output form of *pwrite.*

4. Write a function *perase* to erase the polynomial *c* (in **Program Exercise 2**) and return the polynomial represented as a circular list to the available space list. In addition, write a function *psub* to compute *c=a-b* and store the result to *c*.
   Requirements:
   a. For *c*, a new space for a node can be created only if *avail* points to NULL.
   b. Output the result of *c* in the output form of *pwrite.*

5. We want to implement a complete linked list system to perform arithmetic on sparse matrices using our linked list representation.
   Requirements:
   a. Create your own sparse matrices pointed by *a* and *b.* Allow users to input the number of rows, the number of columns, the number of nonzero terms, and the value and location of each nonzero term in row major form, for each sparse matrix.
   b. Write a function *mmult*, to compute *c=a*b.*
   c. Print out *c* (in the form of array for sparse matrices in Chapter 2)