MATH 895, Assignment 5, Summer 2017

Instructor: Michael Monagan

Please hand in the assignment by 8:30pm on Thursday July 6th.
Late Penalty -20% off for up to 30 hours late. Zero after that.
Please submit a printout of a Maple worksheet containing Maple code and output.

Download and read the paper "Sparse Polynomial Arithmetic" by Stephen Johnson. Notice that Johnson's paper assumes univariate polynomials only. One can map a multivariate polynomial $f(x, y, z)$ into a univariate polynomial $g(x)$ by means of the Kronecker substitution: $g := subs(y = x^j, z = x^k, f)$ for sufficiently large j, k in such a way that one can recover $f(x, y, z)$ from $g(x)$.

Let $A, B \in \mathbb{Q}[x, y, z, ...]$ and let $C = A \times B$ and let Q be the quotient of C divided B. Represent a polynomial as a Maple list of terms sorted in descending *graded lexicographical* order. Represent each term in the form $[c, e]$ where $c \in \mathbb{Q}$ is a coefficient and e, the exponent vector, is encoded as an integer as follows: the monomial $x^i y^j z^k$ with exponent vector $[i, j, k]$ would be represented as the integer $e = (i + j + k)B^3 + iB^2 + jB + k$ where $B = 2^L$ bounds the total degree d of any monomial that appears in the multiplication/division algorithm.

Implement the following Maple procedures where X is a list of variables.

```
A := SDMP2Maple(a,X,B);
a := Maple2SDMP(A,X,B);
```

E.g. `A := SDMP2Maple(a,[x,y,z],B)` converts a Maple polynomial $a(x, y, z)$ into the SDMP data structure and `Maple2SDMP(A,[x,y,z])` converts it back. Note, to convert an integer E to base B in Maple use `convert(E,base,B);` Note, to sort the terms in a polynomial you can use the `sort` command. Now implement the following four algorithms:

1 Classical merge algorithm for multiplication: $f \times g = ((f_1 \times g + f_2 \times g) + f_3 \times g) + \dots$.

2 Johnson's heap multiplication algorithm: $f \times g = \sum_{i=1}^{\#f} f_i \times g$.

3 Classical merge algorithm for division: $f \div g = (((f - q_1 \times g) - q_2 \times g) - q_3 \times g) - \dots$.

4 Johnson's quotient heap division algorithm: $f \div g = f - \sum_{i=1}^{\#q} q_i \times g$.

Execute your algorithms on the following sparse problem

```
  > X := [u,v,w,x,y,z];
  > a := randpoly(X,degree=10,terms=2500):
  > b := randpoly(X,degree=5,terms=10):
  > c := expand(a*b):
  > nops(a), nops(b), nops(c);
                                           2479, 10, 19172

  > d := degree(a)+degree(b);
                                                 15
  > B := 16:
  > A := Maple2SDMP(a,X,B):
  > B := Maple2SDMP(b,X,B); # show your data structure for this one
  > C := Maple2SDMP(c,X,B):
  > H := MULTIPLY(A,B): evalb(H=C);
  > H := MULTIPLY(B,A): evalb(H=C);
  > Q := DIVIDE(C,A); evalb(Q=B); # show output for Q
  > Q := DIVIDE(C,B): evalb(Q=A);
```

Compute and print (i) N = the number of monomial comparisons each algorithm makes, (ii) M = the number of coefficient multiplications + divisions each algorithm makes and (iii) the quantity $S = N/M$ which measures the monomial comparisons relative to the coefficient arithmetic cost. Now, what is the theoretical number of monomial comparisons that the two algorithms should make for these inputs? Compare these with the actual values.

For the heap operations you may use Maple's `heap` package. See `?heap`.

To count the number of comparisons done in the heap insertions and extractions, use a global variable like this:

```
  > less := proc(a,b) global N; N := N+1; evalb( a[2]<b[2] ) end;
  > H := heap[new](less);
  > N := 0;
```

If you feel inspired, parameterize your multiplication and division routines by the coefficient domain R by using a simple table of Maple procedures and constants. For example, for $R = \mathbb{Z}$ use

```
> R[0] := 0; R[1] := 1;
> R[`*`] := proc(a,b) a*b end;
> R[`+`] := proc(a,b) a+b end;
> R[`-`] := proc(a,b) a-b end;
> R[neg] := proc(a) -a end;
> R[div] := proc(a,b,q) evalb( irem(a,b,q)=0 ) end;
```