```
> (* Mantej Sokhi *)
> restart:
```

Q2: MULTIPLICATION USING JOHNSONS HEAP:

```
> MAPLE2SDMP := proc(a,X::list(name))

  local C,M,n,t,A,E,i,m:

  if not type(a,polynom(rational,X)) then
     error "BAD INPUT"
  fi:

  C := [coeffs(a,X,'M')]:
  M := [M]:
  n := nops(X):
  t := nops(M):
  E := [seq([seq(degree(m,X[i]),i=1..n)],m in M)]:
  A := [seq([C[i],E[i]],i=1..t)]:

  sort(A,proc(x,y) GREATER(x[2],y[2]) end):
  end proc:
> SDMP2MAPLE := proc(A::list([rational,list(nonnegint)]),X::list
  (name))

  local t,i,n:

  n := nops(X):
  add(t[1]*mul(X[i]^t[2][i],i=1..n),t in A):
  end proc:
> GREATER := proc(a::list(nonnegint),b::list(nonnegint))

  local dega,degb,n,i,z:
  n := nops(a):
  if nops(b)<>n then:
     error "EXP VECTORS OF DIFF SIZE":
  fi:

  dega := add(z,z in a):
  degb := add(z,z in b):

  if dega>degb then:
      return true:
  elif dega<degb then:
      return false:
  fi:
```

```
  for i to nops(a) do:
     if a[i]>b[i] then:
        return true:
     elif a[i]<b[i] then:
        return false:
      fi:
  od:
  false:
  end proc:
> GRT := proc(a,b)

  global monComp:
  monComp := monComp+1:

  evalb(GREATER(b[3],a[3])):
  end proc:
> X := [u,v,w,x,y,z]:
  a := randpoly(X,degree=10,terms=5000):
  b := randpoly(X,degree=10,terms=50):
  A := MAPLE2SDMP(a,X):
  B := MAPLE2SDMP(b,X):
  c := expand(a*b):
  C := MAPLE2SDMP(c,X):
> nops(a);
  nops(b);
  nops(c);

                                4966
                                  49
                              127191                                  (1)
> d := degree(a)+degree(b):
  d;

                                  20                                  (2)
> HEAPMUL := proc(A,B)

  local H,i,j,h,m,n,hashTable,extVal,extI,extJ,c,INSERT,ratio,k:

  H := heap[new](GRT):
  hashTable := table():
  m := nops(A):
  n := nops(B):
  h := table():
```

```
k := 1:

global monComp, coeffMul:
monComp := 0:
coeffMul := 0:

INSERT := proc(i::posint,j::posint)

if (i<=m) and (j<=n) then:
   heap[insert]([i,j,A[i][2]+B[j][2]],H):
f i :

end proc:

INSERT(1,1):

while not heap[empty](H) do:
   extVal := heap[extract](H):
   extI := extVal[1]:
   extJ := extVal[2]:
   c := A[extI][1]*B[extJ][1]:
   coeffMul := coeffMul+1:

   if extJ=1 and extI<m then:
      INSERT(extI+1,extJ):
    fi:

   if extJ<n then:
      INSERT(extI,extJ+1):
    fi:

   while not heap[empty](H) and extVal[3]=heap[max](H)[3] do:
      extVal := heap[extract](H):
      extI := extVal[1]:
      extJ := extVal[2]:
      c := c+(A[extI][1]*B[extJ][1]):
      coeffMul := coeffMul+1:

      if extJ=1 and extI<m then:
         INSERT(extI+1,extJ):
       fi:

      if extJ<n then:
         INSERT(extI,extJ+1):
       fi:
```

```
    od:
    if c<>0 then:
       h[k] := [c,extVal[3]]:
       k := k+1:
    fi:
  od:

  h := convert(h,list):

  ratio := evalf(monComp/coeffMul,5):
  return h,monComp,coeffMul,ratio:
  end proc:
> h1,monComp,coeffMul,ratio := HEAPMUL(A,B):
  evalb(h1=C);
```

true **(3)**

```
> h2,mC2,cM2,r2 := HEAPMUL(B,A):
  evalb(h2=C);
```

true **(4)**

```
> printf("TOTAL MONOMIAL COMP. -> %a.\nTOTAL COEFF. MUL. COMP. ->
  %a.\nRATIO -> %a.\n",monComp,coeffMul,ratio);

TOTAL MONOMIAL COMP. -> 2273569.
TOTAL COEFF. MUL. COMP. -> 243334.
RATIO -> 15.049.
```