Poly Addition:

```
Algorithm PolyAdd(p,q,r)
{  if p!=Null && q!=Null  then
      { i:=p, j:=q, r:=k:=Null
      while( i!=Null && j:=Null) do
      {  new:=Avail, Avail:=Link[Avail]
         if exp[i]=exp[j] then {
cof[new]:=cof[i]+cof[j]

exp[new]:=exp[i]
                                          i:=link[i],
j:=link[j] }
         else if(exp[i]>exp[j]  then {
cof[new]:=cof[i]

exp[new]:=exp[i]

i:=link[i] }
         else { cof[new]:=cof[j]
                exp[new]:=exp[j]
                 j:=link[j] }
```

```
If r=Null then { r:=k:=new}
  .  else { link[k]:=new, k:=new }
  . }// end of while loop
  .  if i=Null then { while(j!=Null) do
  .                              { new:=avail,
    avail:=link[avail]
  .                                 cof[new]:=cof[j],
    exp[new]:=exp[j]
  .                                 j:=link[j] }
  .                        }
  . else {while(i!=Null) do
  .                              { new:=avail,
    avail:=link[avail]
  .                                 cof[new]:=cof[i],
    exp[new]:=exp[i]
  .                                 i:=link[i] }
  .           }
  . link[k]:=Null;  }//end of algorithm
```

Conversion of Infix to Postfix

- **Algorithm Con-Post (Q,P)**

- { // Q is an infix expn and P is the post fix expn
- 1. PUSH “(“ onto stack and add “)” to the end of Q
- 2. Scan Q from left to right and repeat steps 3 to 6 for each element of Q until the stack is empty
- 3. If an operand is encountered, add it to P
- 4. if a “(“ is encountered then, PUSH it into stack
- 5. if an operator ‘X’ is encountered then
- a) Repeatedly POP from stack and add to P each operator which has the same precedence as or higher precedence than ‘X’
- b) Add ‘X’ to stack
- 6. If a “)” is encountered then
- a) Repeatedly POP from stack and add to P each operator until a “(“ is encountered
- b) Remove the “(“ from stack
- }

Evaluation of Postfix Expression

- **Algorithm Evalu (P)**
- **{ // P is the given post fix expression and stack is used to store the operands**
- **1. Add a “)“ at the end of P**
- **2. Scan P from left to right and repeat steps 3 to 4 for each element of P until the “)” is encountered**
- **3. If an operand is encountered, PUSH it into stack**
- **4. if an operator ‘X’ is encountered then**
- **a) POP the two top elements from stack , where A is the first top and B is the next top element**
- **b) Evaluate B ‘X’ A**
- **c) PUSH the result of step b into stack**
- **5. set the value equal to the top element of stack**
- **}**