```
Polynomial AddPolynomial (Polynomial P1, Polynomial P2)
{     P3 =(Polynomial)malloc(sizeof(Poly))
      P3 -> HighPower = max (P1->HighPower, P2->HighPowe
      for (int i=0 , i<P3->HighPower , i++) {
          P3 -> Coefficient [i] = P1->Coefficient [i] + P2->Coef
               [i] }
      return P3
Polynomial MultPolynomial (Polynomial P1 . Polynomial P2)
      P3 =(Polynomial)malloc (sizeof (Poly))
      P3 -> High Power = P1->HighPower + P2-> HighPower
      for (int i=0 ; i<=P1->High Power, i++) {
          for (int j=0 ; j<=P2->HighPower ; j++) {
          P3->Coefficient [i+j] = P1->Coefficient [i]+P2->Coefficien

      }

   Polynomial AddPoly (Polynomial P1, Polynomial P2)
   P3 =(PtrToNode)malloc(sizeof(PolyNode));
   while (P1!=NULL && P2!=NULL) {
       if (P1->Exp == P2->Exp) {
            if (P1->Coeff + P2->Coeff == 0) {
                 P1 = P1->next
                 P2 = P2->next
            }
            else {
```

```
else {
        P3->Coeff = P1->Coeff + P2->Coeff
        P3->Exp = P1->Exp
        P3 = P3->next
        P1 = P1->next
        P2 = P2->next
    if (P1->Exp > P2->Exp) {
      P3->Exp = P2->Exp
      P3->Coeff = P2->Coeff.
       P3 = P3->next
       P2 = P2->next }
    if (P1->Exp < P2->Exp) {
      P3->Exp = P1->Exp
      P3->Coeff = P1->Coeff
       P3 = P3->next
       P2 = P2->next }
  }
  else {
     if (P1 == NULL) {
      P3->Exp = P2->Exp
       P2->Coeff = P2->Coeff
     if (P2 == NULL) {
      P3->Exp = P1->EXP
       P2->Coeff = P1->Coeff }
    else {
      printf("error")
```

```
            [illegible]
            return 1 }
        }
        return P3

Polynomial MultPoly (Polynomial P1, Polynomial P2)
    P3 = (PtrToNode)malloc(sizeof(PolyNode)) < if (P1==N
                                                  ||P2==
                                               reture er
    while (P1!=NULL&&P2!=NULL) {
     temp=(PtrToNode)malloc(sizeof(PolyNode))
        while(P2!=NULL) {
         temp->Exp=P1->Exp+P2->Exp.
         temp->Coeff = P1->Coeff x P2->Coeff }
        AddPoly (P3, temp)
        free(temp)
        else {
            P1=P1->next }
    else {
        return P3 }
```