ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

	ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ	**Τ. Σελλής - Ν. Κοζύρης**
ΑΝΟΙΞΗ 2002	**ΑΣΚΗΣΗ #2**	**Ημερ. Παράδοσης: 29/3/02**

ΕΡΩΤΗΜΑ 1

Θεωρούμε την παράσταση ενός πολυωνύμου μιας μεταβλητής με τη βοήθεια συνδεδεμένης λίστας. Κάθε στοιχείο της λίστας αντιστοιχεί σε ένα όρο του πολυωνύμου και περιλαμβάνει τρία πεδία: συντελεστή (διάφορο του μηδενός), εκθέτη και δείκτη προς το επόμενο στοιχείο. Υποθέτουμε ότι η λίστα είναι διατεταγμένη κατά φθίνουσες τιμές του εκθέτη. Να γραφτούν δύο συναρτήσεις AddPoly και MultPoly (σε C, Pascal ή ψευδοκώδικα) οι οποίες να δέχονται ως παραμέτρους δύο πολυώνυμα με τη μορφή συνδεδεμένων λιστών και να δημιουργούν μια νέα λίστα η οποία αντιπροσωπεύει το άθροισμα και το γινόμενο των δύο πολυωνύμων αντίστοιχα

ΕΡΩΤΗΜΑ 2

Να υλοποιηθεί η συνάρτηση διάσχισης δυαδικού δέντρου Inorder χωρίς αναδρομική κλήση συναρτήσεων.

ΕΡΩΤΗΜΑ 1

Για τους όρους του πολυωνύμου προτείνεται η χρησιμοποίηση της ακόλουθης δομής:

```
typedef struct pterm{
      double coef; //συντελεστής
      unsigned int exp; //εκθέτης
      struct pterm* next; // δείκτης σε επόμενο όρο
      } PolyTerm;
```

Παρακάτω δίνονται δυνατές υλοποιήσεις για τις συναρτήσεις AddPoly και MultPoly. Η AddPoly προσθέτει δύο πολυώνυμα polyA και polyB, και επιστρέφει δείκτη στο αποτέλεσμα, ενώ η MultPoly πολλαπλασιάζει δύο πολυώνυμα polyA και polyB και επιστρέφει τη διεύθυνση του γινομένου.

```
PolyTerm* AddPoly(PolyTerm* polyA,PolyTerm* polyB)
{
      PolyTerm* polyC=NULL;
      PolyTerm *tmpA, *tmpB, *endC;
      PolyTerm* res;
      int finished;

      tmpA=polyA; //tmpA: διατρέχει το πρώτο πολυώνυμο
      tmpB=polyB; //tmpB: διατρέχει το δεύτερο πολυώνυμο
      finished=(tmpA==NULL)&&(tmpB==NULL);

      while(!finished)
      {
            finished=1;
            res=(PolyTerm*)malloc(sizeof(PolyTerm)); // res:νέος όρος
            res->next=NULL; // κάθε νέος όρος εισάγεται στο τέλος
            //υπολογισμός συντελεστή και εκθέτη νέου όρου res
            if((tmpB==NULL)||((tmpA!=NULL)&&(tmpB!=NULL)&&(tmpA->exp>tmpB->exp)))
            {
                  res->exp=tmpA->exp;
                  res->coef=tmpA->coef;
                  tmpA=tmpA->next;
            }
            else if((tmpA==NULL)||((tmpA!=NULL)&&(tmpB!=NULL)&&(tmpB->exp>tmpA-
>exp)))
            {
                  res->exp=tmpB->exp;
                  res->coef=tmpB->coef;
                  tmpB=tmpB->next;
            }
            else if((tmpA!=NULL)&&(tmpB!=NULL)&&(tmpA->exp==tmpB->exp))
            {
                  res->exp=tmpA->exp;
                  res->coef=tmpA->coef+tmpB->coef;
                  tmpA=tmpA->next;
                  tmpB=tmpB->next;
            }

            // ενημέρωση λίστας
            // polyC: αποτέλεσμα (άθροισμα)
            // endC: δείκτης σε τελευταίο όρο αποτελέσματος
            if(res->coef!=0)
            {
                  if(polyC==NULL)
                        polyC=endC=res;
                  else
                  {
                        endC->next=res;
                        endC=res;
                  }
            }
```

```
            finished=(tmpA==NULL)&&(tmpB==NULL);
      }

      return polyC;
}

PolyTerm* MultPoly(PolyTerm* polyA,PolyTerm* polyB)
{
      PolyTerm *tmpA,*tmpB,*polyC,*startC,*endC,*res;

      tmpA=polyA; // tmpA: διατρέχει πρώτο πολυώνυμο
      polyC=NULL;

      while(tmpA!=NULL)
      {
            tmpB=polyB; // tmpB: διατρέχει δεύτερο πολυώνυμο
            startC=endC=NULL; //startC,endC:αρχή,τέλος μερικού
                              //αθροίσματος
            while(tmpB!=NULL)
            {
                  res=(PolyTerm*)malloc(sizeof(PolyTerm));
                  res->coef=tmpA->coef*tmpB->coef;
                  res->exp=tmpA->exp+tmpB->exp;
                  res->next=NULL;
                  if(startC==NULL)
                        startC=endC=res;
                  else
                  {
                        endC->next=res;
                        endC=res;
                  }
                  tmpB=tmpB->next;
            }
            // polyC: γινόμενο
            polyC=AddPoly(polyC,startC);
            tmpA=tmpA->next;
      }

      return polyC;
}
```

ΕΡΩΤΗΜΑ 2

Για τη διάσχιση του δυαδικού δέντρου χωρίς αναδρομική κλίση συναρτήσεων θα χρησιμοποιήσουμε τη δομή της στοίβας. Θεωρούμε γνωστές τις συναρτήσεις εισαγωγής (push) και εξαγωγής (pop) στη στοίβα και ελέγχου αν η στοίβα έχει ακόμα στοιχεία (nonempty). Μια ενδεικτική ενδοδιατεταγμένη διάσχιση υλοποιείται με τη συνάρτηση που δίνεται στη συνέχεια:

```
void InOrder(treenode *root)
{
  treenode *h;
  stack *p;

  h=root;
  do{
      while(h!=0){push(p,h);h=h->left;}
      do{pop(p,h);visit(h);}while(h->right==0);
      h=h->right;
  }while(nonempty(p));

}
```