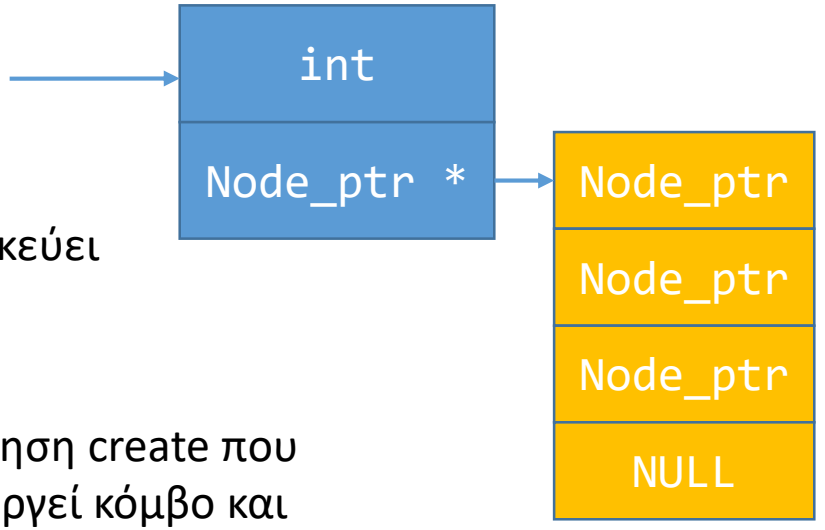
Διαδικαστικός προγραμματισμός

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
    int data;
    struct node **next;

} Node ;

typedef Node * Node_ptr;
typedef Node * Multitree;

Node_ptr create(int a );
Node_ptr connect(Node_ptr g1_ptr, Node_ptr g2_ptr);

void report(Multitree t);
int sum(Multitree t);
```



Τύπος Node κόμβου που αποθηκεύει ακέραιο και διεύθυνση πίνακα συνδέσεων.

Συνάρτηση create που δημιουργεί κόμβο και επιστρέφει δείκτη στο νέο κόμβο

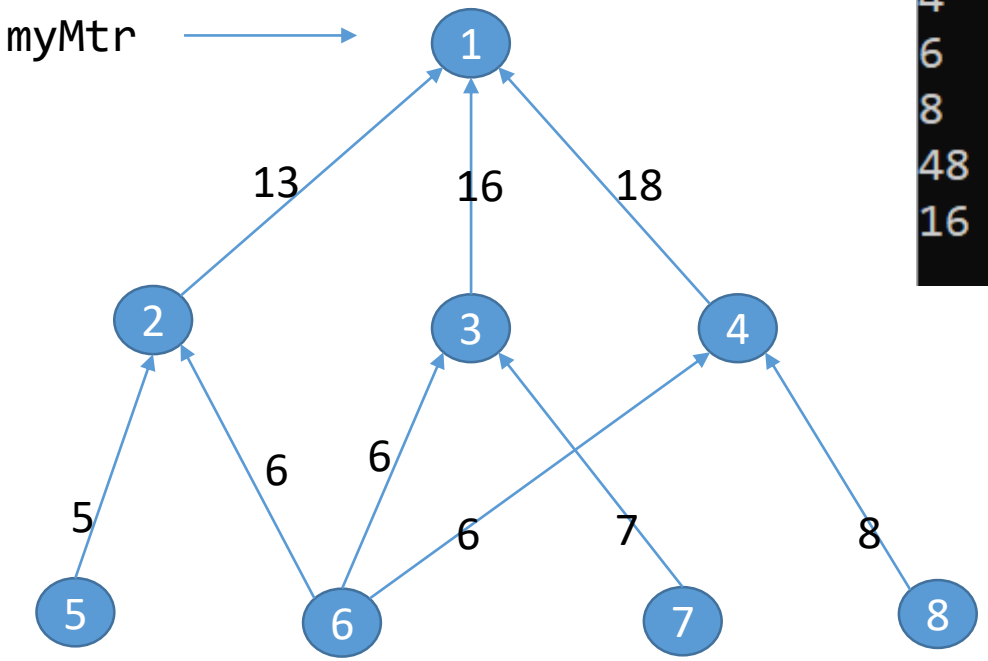
Συνάρτηση connect για σύνδεση δύο κόμβων

Συνάρτηση report διατρέχει την κατασκευή και τυπώνει την τιμή του κάθε κόμβου

Συνάρτηση sum επιστρέφει το άθροισμα της τιμής του κάθε κόμβου και με το σύνολο των τιμών που έρχονται από τους απογόνους του (διαφορετικό από το συνολικό άθροισμα!).

```c
int main()
{
    Multitree myMtr  = NULL;
    Node_ptr g2_ptr, g3_ptr, g4_ptr, g5_ptr;
    Node_ptr g6_ptr, g7_ptr, g8_ptr;
    myMtr = create(1);

    g2_ptr = create(2);
    connect(myMtr, g2_ptr);
    g3_ptr = create(3);
    connect(myMtr, g3_ptr);
    g4_ptr = create(4);
    connect(myMtr, g4_ptr);
    connect(g2_ptr, g5_ptr=create(5));
    connect(g2_ptr, g6_ptr=create(6));
    connect(g3_ptr, g6_ptr);
    connect(g3_ptr, g7_ptr=create(7));
    connect(g4_ptr, g6_ptr);
    connect(g4_ptr, g8_ptr=create(8));

    report(myMtr);

    printf("%d\n", sum(myMtr));

    printf("%d\n", sum(g3_ptr));

    return 0;
}
```

```
1
2
5
6
3
6
7
4
6
8
48
16
```



Υλοποίηση της συγκεκριμένης διασύνδεσης με χρήση συνάρτησης connect την οποία κατασκευάζουμε

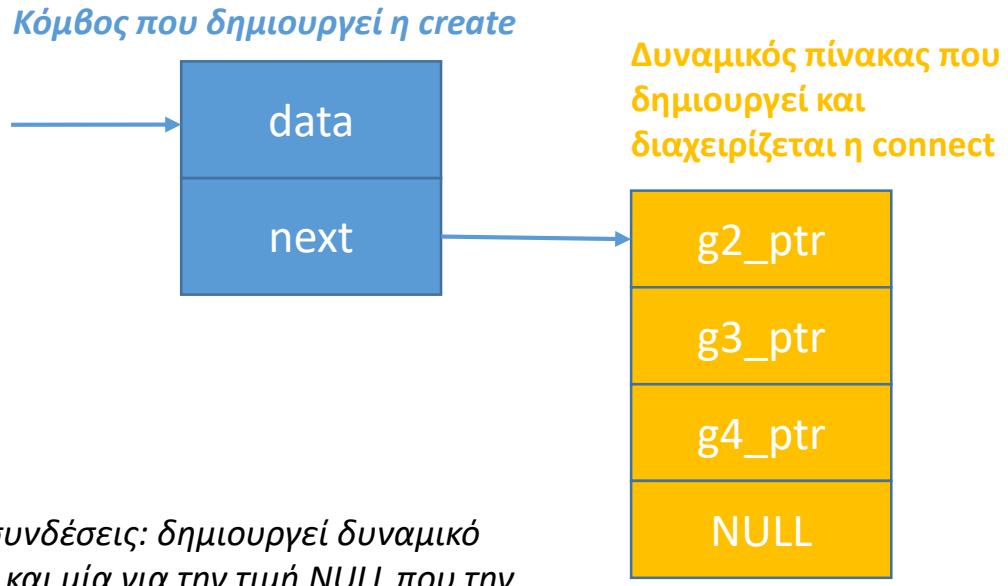
```
Node_ptr create(int a) {
    Node_ptr g_ptr = malloc(sizeof (Node));
    if (g_ptr==NULL) return NULL;

    g_ptr->data = a;
    g_ptr->next = NULL;

    return g_ptr;
}
```

Δημιουργία κόμβου και επιστροφή δείκτη σε αυτόν.

Δεν δεσμεύει χώρο για τον δυναμικό πίνακα συνδέσεων, αυτό γίνεται στην connect

```
Node_ptr connect(Node_ptr g1_ptr, Node_ptr g2_ptr) {

    if (g1_ptr==NULL) return NULL ;

    if (g1_ptr->next == NULL) {
        g1_ptr->next = malloc ( 2 * sizeof (Node_ptr) ) ;
        if (g1_ptr->next == NULL) return NULL;
        (g1_ptr->next)[0] = g2_ptr;
        (g1_ptr->next)[1] = NULL;

        return g1_ptr;
    }
```

Περίπτωση όπου δεν υπάρχουν προηγούμενες συνδέσεις: δημιουργεί δυναμικό πίνακα δύο θέσεων, μία για την πρώτη σύνδεση και μία για την τιμή NULL που την χρησιμοποιούμε για να δηλώσουμε το τέλος του πίνακα.




```
    {
        int count = 0;
        while((g1_ptr->next)[count]!=NULL) count++;      /* one-line while */

        {
          Node_ptr * temp_ptr= realloc(g1_ptr->next, (count+2)*sizeof (Node_ptr) );
          if (temp_ptr == NULL) return NULL;              /* can handle in better way */
          g1_ptr->next = temp_ptr;
        }
        (g1_ptr->next)[count] = g2_ptr;
        (g1_ptr->next)[count+1] = NULL;
    }

    return g1_ptr;
}
```

Περίπτωση όπου υπάρχουν προηγούμενες συνδέσεις: μετράει τα στοιχεία του δυναμικού πίνακα, αυξάνει το μέγεθος του κατά μία θέση, εγγράφει τη νέα σύνδεση και τοποθετεί NULL για να δηλώσει το τέλος.

```
void report(Multitree t) {

    if (t == NULL) return ;          /* empty, do nothing */
    printf("%d\n", t->data);         /* print value in node */

    if (t->next == NULL) return;     /* if no children, return */

    {                           /* there exist children, recursively visit each one */
    int i =0 ;
    while ((t->next)[i] != NULL) {
        report((t->next)[i]);
        i++;
       }
    }
    return ;
}
```

```
int sum(Multitree t) {
    int tempsum;
    if (t == NULL) return 0;
    if (t->next == NULL) return t->data;

    {
    int i = 0 ;
    tempsum = t->data;
    while ((t->next)[i] != NULL) {
        tempsum += sum((t->next)[i]);
        i++;
       }
    }

    return tempsum;
}
```

Επεκτείνει τη λογική της report για να κάνει έναν υπολογισμό. *Προσέξτε ότι ένας κόμβος που είναι απόγονος πολλών, εδώ λαμβάνεται υπόψη πολλές φορές.*

Συνδυάζει επανάληψη και αναδρομή.

QUIZ 1: Μπορείτε να γράψετε μια tail recursive έκδοση της συνάρτησης sum ?

QUIZ 2: Μπορείτε να τροποποιήσετε το πρόγραμμα ώστε να μην λαμβάνεται υπόψη πάνω από μία φορές κάθε κόμβος στο άθροισμα;
A) με οποιαδήποτε τροποποίηση
B) με τροποποίηση ΜΟΝΟ της sum, χωρίς αλλαγές στους τύπους.

Ανεβάστε τη λύση στο eclass>εργασίες>QUIZ1+2 (*με ουδέν βαθμολογικό όφελος*)