ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

9η Εβδομάδα: Διασυνδεδεμένες Λίστες

Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Η δήλωση typedef στη C (συν.)

- Πολλές φορές **βαφτίζουμε** μία δομή ταυτόχρονα με τον **ορισμό** της, όπως στο παράδειγμα:

```
typedef struct node {
  int data;
  struct node *next;
} NODE;
```

- Είναι καλή συνήθεια να χρησιμοποιούμε ως όνομα στο `typedef` το ίδιο όνομα της δομής αλλά με κεφαλαία γράμματα.
- Γιατί να χρησιμοποιούμε τη δήλωση `typedef` ;
  1. Γιατί δημιουργεί πιο αναγνώσιμο και πιο κατανοητό κώδικα . Το να δηλώσεις `NODE *ptr;` είναι πιο κατανοητό από το να δηλώσεις ένα δείκτη σε μία πολύπλοκη δομή (`struct node *ptr;`).
  2. Ο κώδικας γίνεται πιο λιτός. Π.χ.

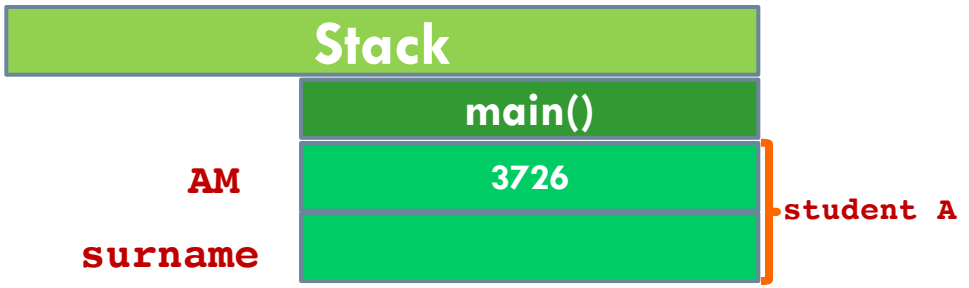
```
ptr = (NODE *)malloc(sizeof(NODE));
```

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
   int AM;
   char surname[50];
};

typedef struct node STUDENT;

void setStudent(STUDENT x);
int main()
{
    STUDENT studentA;
    studentA.AM=3726;
    setStudent(studentA);
    printf ("%d %s",studentA.AM, ,studentA.surname);
    return 0;
}

void setStudent(STUDENT x){
    x.AM=10;
    strcpy(x.surname,"FIDAS");

    return;
}
```

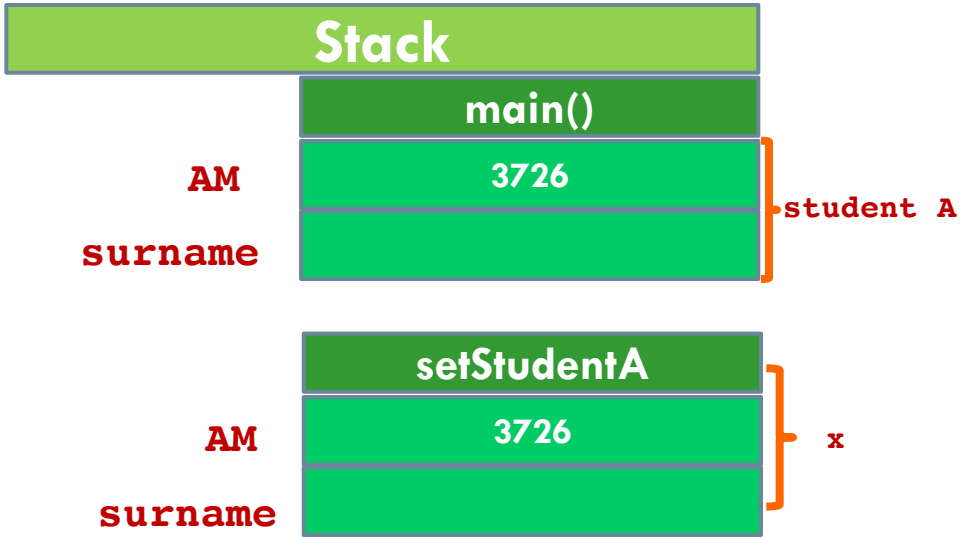
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
   int AM;
   char surname[50];
};

typedef struct node STUDENT;

void setStudent(STUDENT x);
int main()
{
     STUDENT studentA;
     studentA.AM=3726;
     setStudent(studentA);
     printf ("%d %s",studentA.AM, ,studentA.surname);
     return 0;
}

void setStudent(STUDENT x){
     x.AM=10;
     strcpy(x.surname,"FIDAS");

     return;
}
```

Τι κάνει το παρακάτω πρόγραμμα;



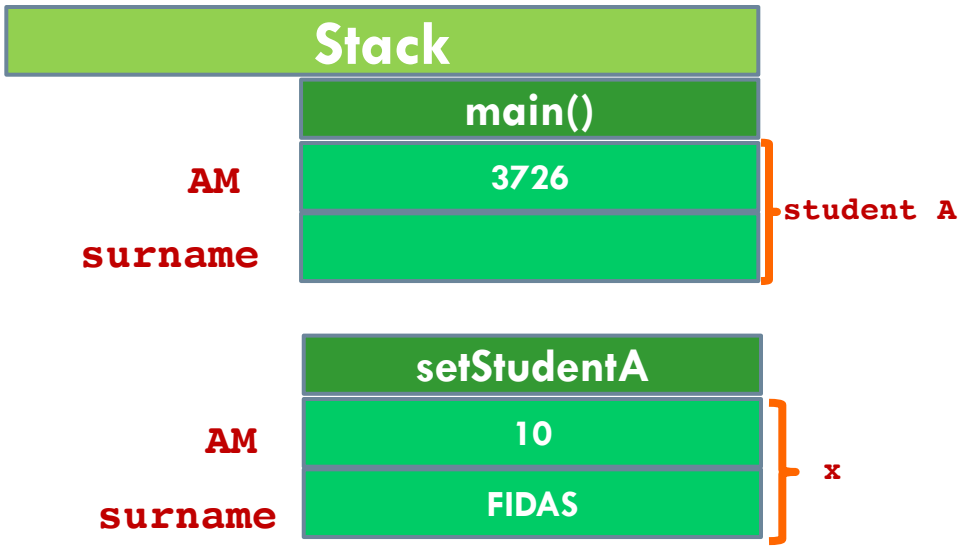
Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
   int AM;
   char surname[50];
};

typedef struct node STUDENT;

void setStudent(STUDENT x);
int main()
{
     STUDENT studentA;
     studentA.AM=3726;
     setStudent(studentA);
     printf ("%d %s",studentA.AM, ,studentA.surname);
     return 0;
}

void setStudent(STUDENT x){
     x.AM=10;
     strcpy(x.surname,"FIDAS");

     return;
}
```



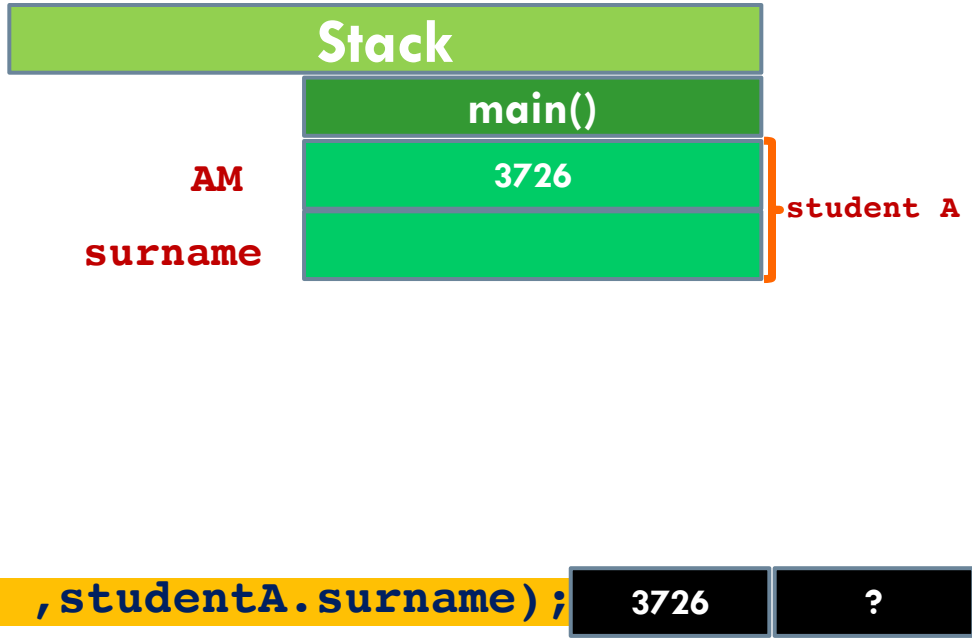
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
   int AM;
   char surname[50];
};

typedef struct node STUDENT;
void setStudent(STUDENT x);
int main()
{
     STUDENT studentA;
     studentA.AM=3726;
     setStudent(studentA);
     printf ("%d %s",studentA.AM, ,studentA.surname);
     return 0;
}

void setStudent(STUDENT x){
     x.AM=10;
     strcpy(x.surname,"FIDAS");

     return;
}
```

Τι κάνει το παρακάτω πρόγραμμα;



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
   int AM;
   char surname[50];
};

typedef struct node STUDENT;

void setStudent(STUDENT x);
int main()
{
     STUDENT studentA;
     studentA.AM=3726;
     setStudent(studentA);
     printf ("%d %s",studentA.AM, ,studentA.surname);
     return 0;
}

void setStudent(STUDENT x){
     x.AM=10;
     strcpy(x.surname,"FIDAS");

     return;
}
```

Τι κάνει το παρακάτω πρόγραμμα;



Με χρήση δείκτη σε δομή

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
    int AM;
    char surname[50];
};

typedef struct node STUDENT;

void setStudent(STUDENT *x);
int main()
{
     STUDENT studentA;
     setStudent(&studentA);
     printf ("%d %s",studentA.AM, ,studentA.surname);
     return 0;
}

void setStudent(STUDENT *x){
     x->AM=10;
     strcpy(x->surname,"FIDAS");

     return;
}
```

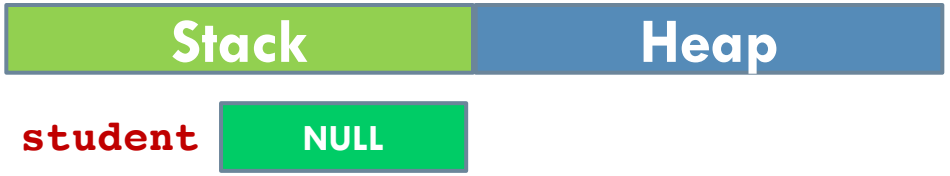
10 FIDAS

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
    int AM;
    char surname[50];
};

typedef struct node STUDENT;
STUDENT setStudent(STUDENT x);
int main()
{
     STUDENT studentA;
     studentA =setStudent(studentA);
     printf ("%d %s",studentA.AM, ,studentA.surname);
     return 0;
}

STUDENT setStudent(STUDENT x){
     x.AM=10;
     strcpy(x.surname,"FIDAS");

     return x;
}
```

Ενναλακτικά

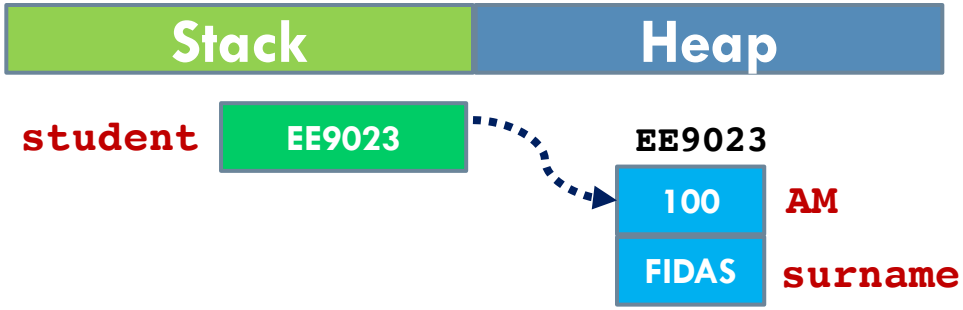
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
    int AM;
    char surname[50];
};

typedef struct node STUDENT;
typedef STUDENT * STUDENT_p;

int main()
{
     STUDENT_p student=NULL;
     student->AM=100;
     strcpy(student->surname,"FIDAS");
     printf ("%d %s",student->AM, ,studentA->surname);
     return 0;
}
```

Τι κάνει το παρακάτω πρόγραμμα;



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
    int AM;
    char surname[50];
};

typedef struct node STUDENT;
typedef STUDENT * STUDENT_p;

int main()
{
     STUDENT_p student=NULL;
     student =(STUDENT_p) malloc(sizeof(STUDENT));
     student->AM=100;
     strcpy(studentB->surname,"FIDAS");
     printf ("%d %s",studentA>AM, ,studentA->surname);
     free(student);
     return 0;
}
```

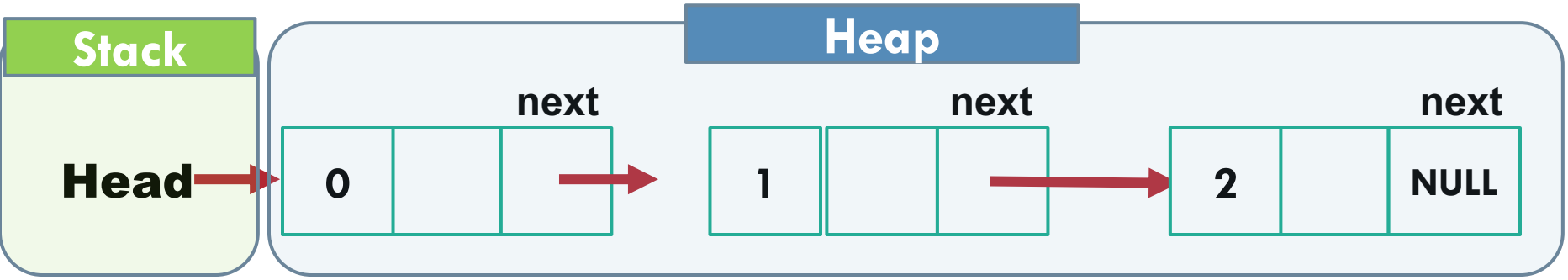
Τι κάνει το παρακάτω πρόγραμμα;



Αυτοαναφορικές δομές και συνδεδεμένες λίστες

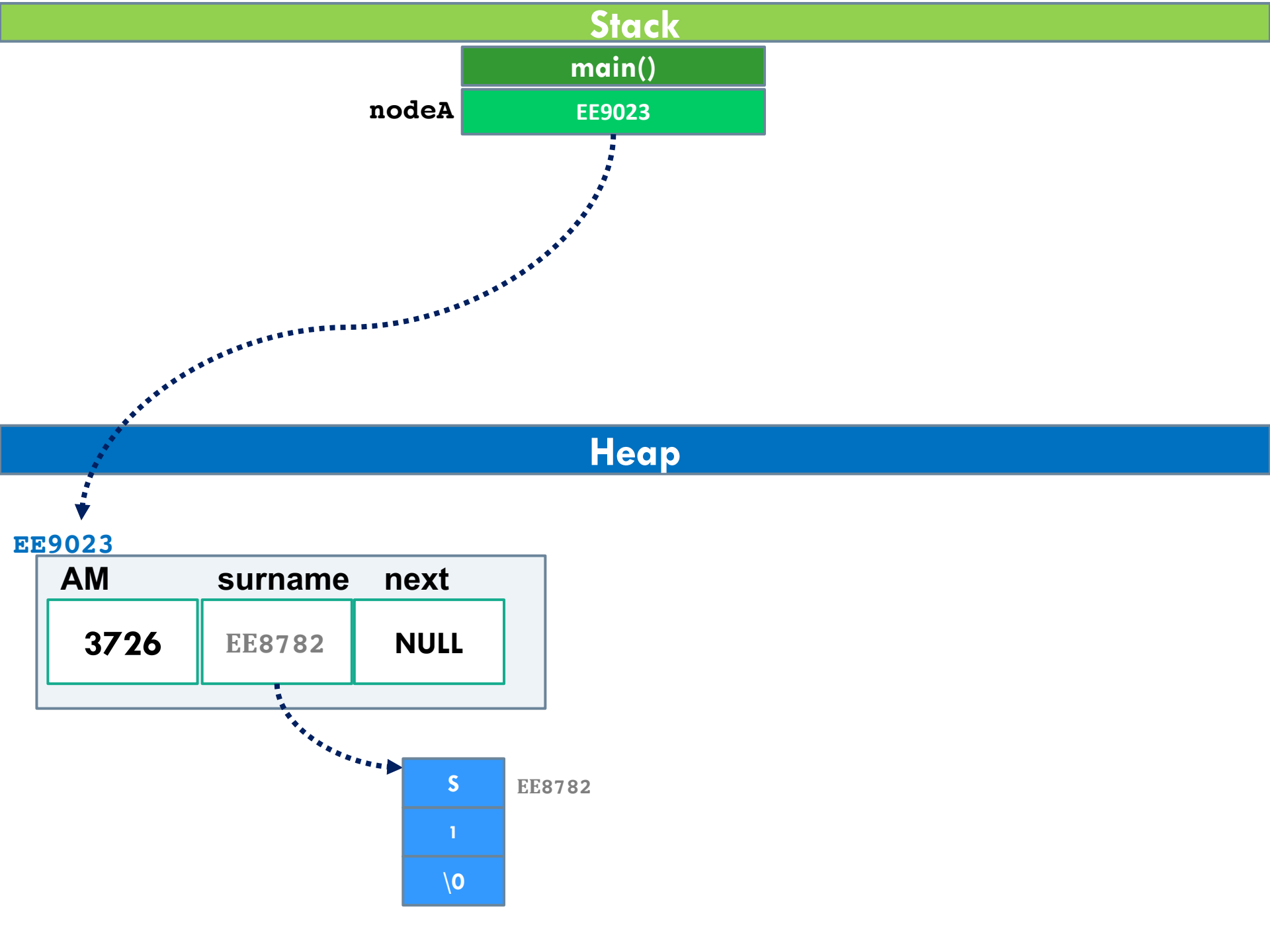
```
typedef struct node {
                int AM;
                char *surname;
                struct node *next;
} STUDENT;
```

Έχοντας λοιπόν καθορίσει τη μορφή ενός κόμβου μπορούμε να φανταστούμε πως θα είναι μία συνδεδεμένη λίστα με κόμβους τύπου struct node που ορίσαμε νωρίτερα:



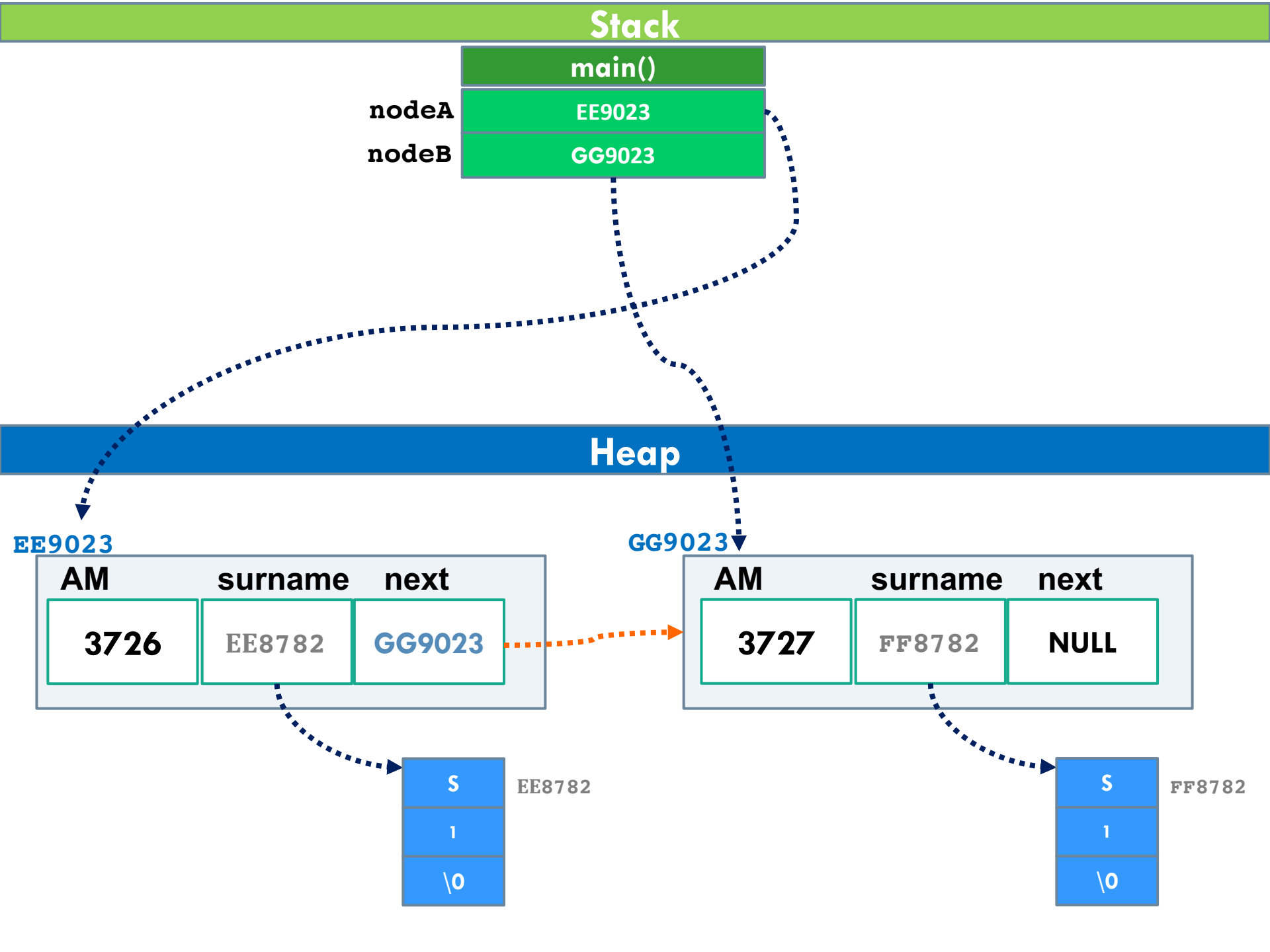
Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
    int AM;
    char *surname;
    struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *nodeA=create_student_node(3726, "s1");
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
         node = (STUDENT *)malloc(sizeof(STUDENT));
         if (node == NULL) { printf("Memory not allocated.\n");
             exit(0);
         }
         else{
           node->AM = AM;
           node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
           strcpy(node->surname, surname);
           node->next = NULL;
           return node;
         }
}
```


Stack
main()
nodeA
EE9023
Heap
EE9023
AM
surname
next
3726
EE8782
NULL
S
1
\0
EE8782

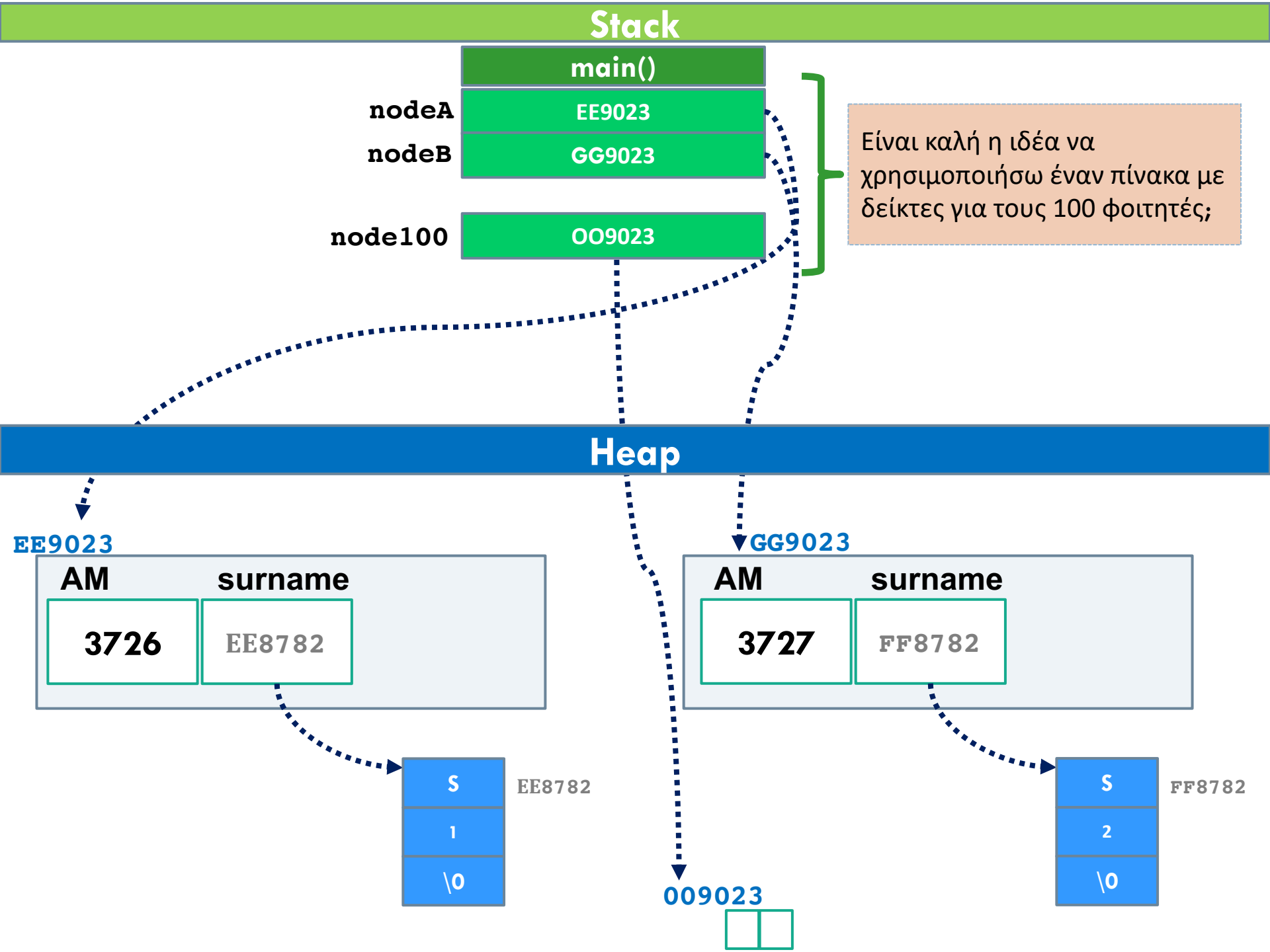
Τι κάνει το παρακάτω πρόγραμμα;

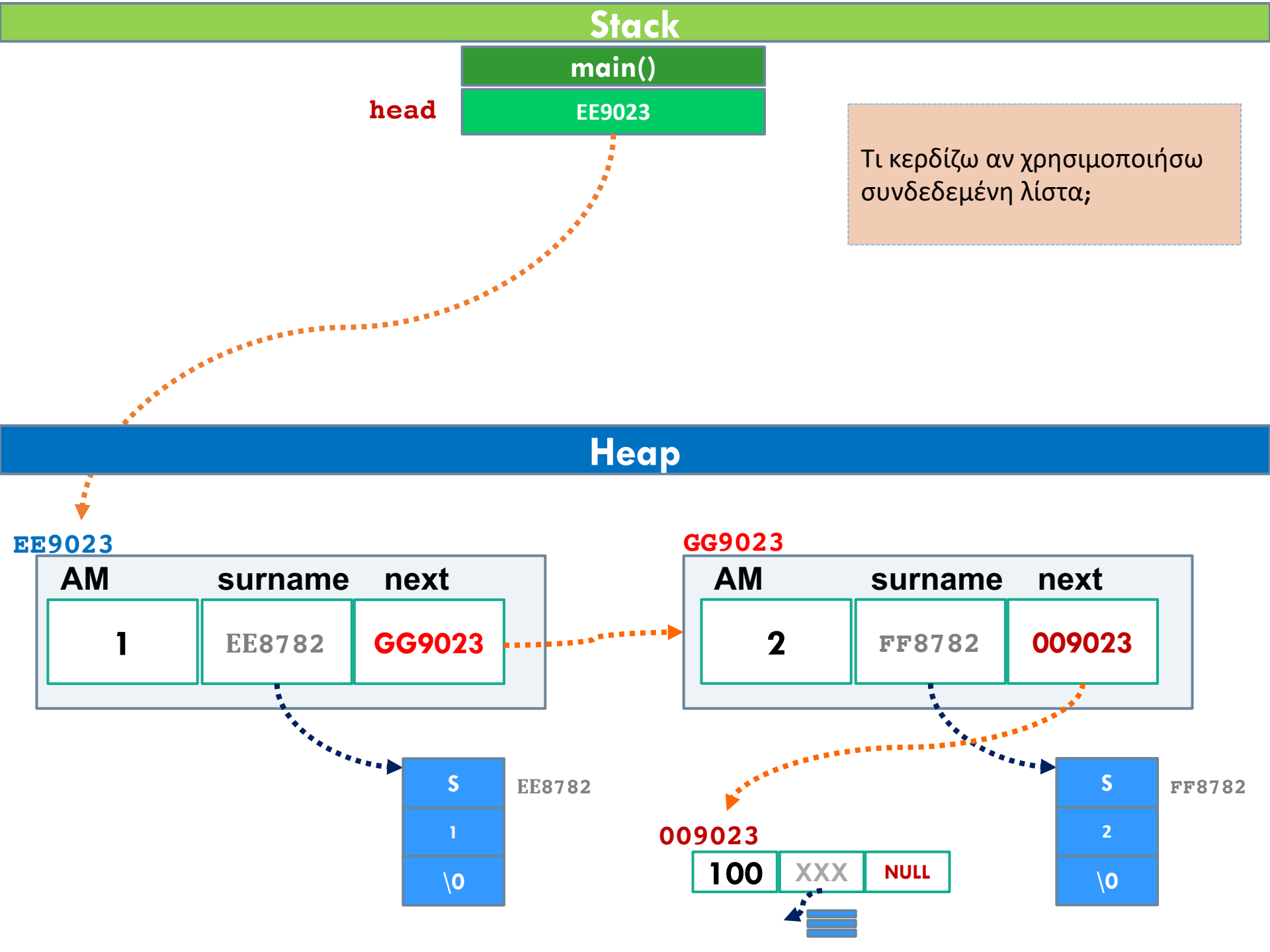
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
    int AM;
    char *surname;
    struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *nodeA=create_student_node(1, "s1");
     STUDENT *nodeB=create_student_node(2, "s2");
     nodeA->next=nodeB;
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
         node = (STUDENT *)malloc(sizeof(STUDENT));
         if (node == NULL) {
             printf("Memory not allocated.\n");
             exit(0);
         }
         else{
           node->AM = AM;
           node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
           strcpy(node->surname, surname);
           node->next = NULL;
           return node;
         }
```

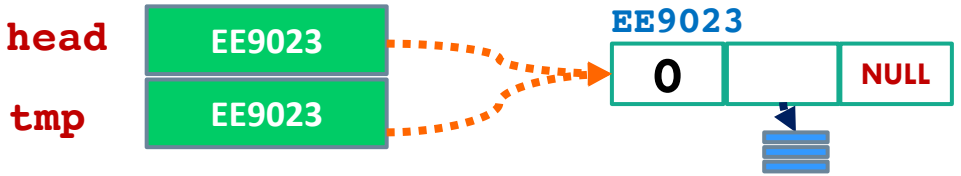

Stack
main()
nodeA
EE9023
nodeB
GG9023
Heap
EE9023
AM
surname
next
3726
EE8782
GG9023
GG9023
AM
surname
next
3727
FF8782
NULL
S
1
\0
EE8782
S
1
\0
FF8782

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
    int AM;
    char *surname;
    struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *nodeA=create_student_node(1, "s1");
     STUDENT *nodeB=create_student_node(2, "s2");
     nodeA->next=nodeB;
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
         node = (STUDENT *)malloc(sizeof(STUDENT));
         if (node == NULL) {
             printf("Memory not allocated.\n");
             exit(0);
         }
         else{
           node->AM = AM;
           node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
           strcpy(node->surname, surname);
           node->next = NULL;
           return node;
```

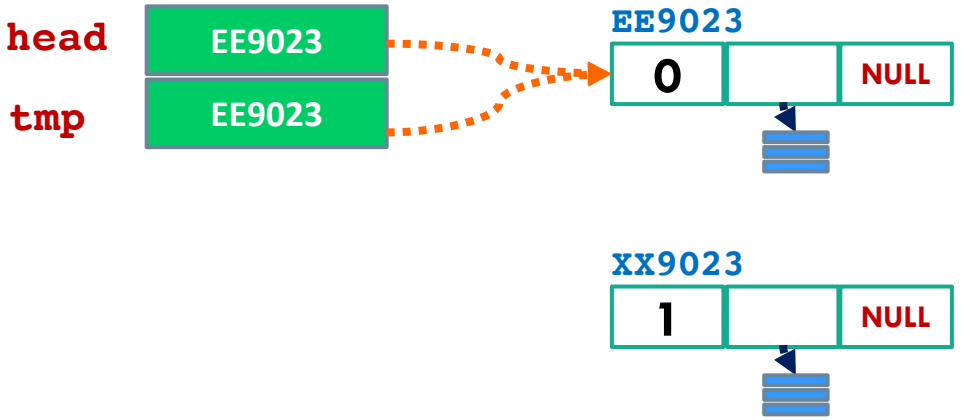
Τι και αν πρέπει να δημιουργήσω 100 κόμβους φοιτητών;


Stack
main()
nodeA
EE9023
nodeB
GG9023
node100
OO9023
Είναι καλή η ιδέα να χρησιμοποιήσω έναν πίνακα με δείκτες για τους 100 φοιτητές;
Heap
EE9023
AM
surname
3726
EE8782
S
1
\0
EE8782
GG9023
AM
surname
3727
FF8782
S
2
\0
FF8782
009023


Stack
main()
head
EE9023
Τι κερδίζω αν χρησιμοποιήσω συνδεδεμένη λίστα;
Heap
EE9023
AM
surname
next
1
EE8782
GG9023
S
1
\0
EE8782
GG9023
AM
surname
next
2
FF8782
009023
S
2
\0
FF8782
009023
100
XXX
NULL

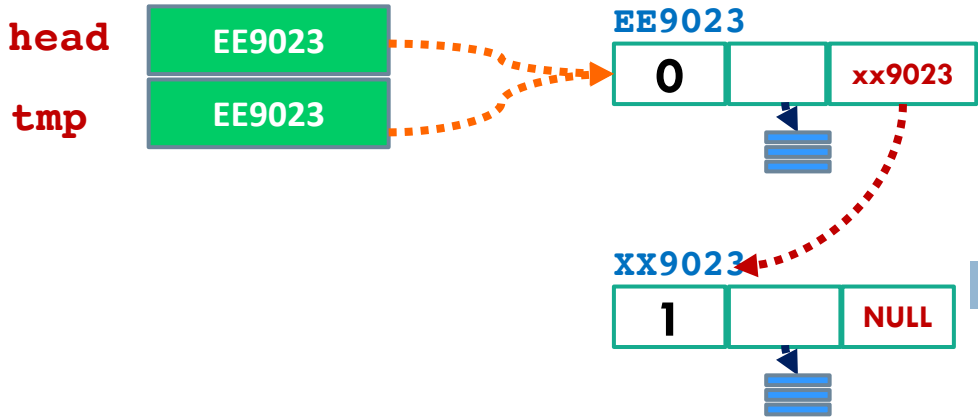



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```

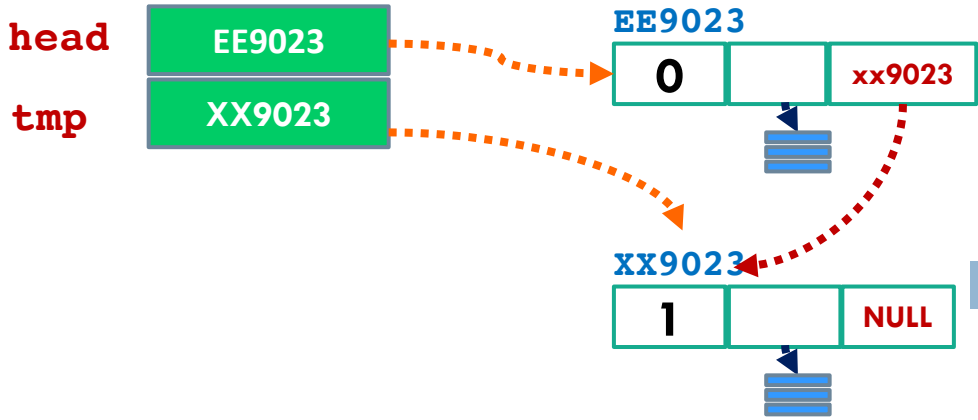



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```

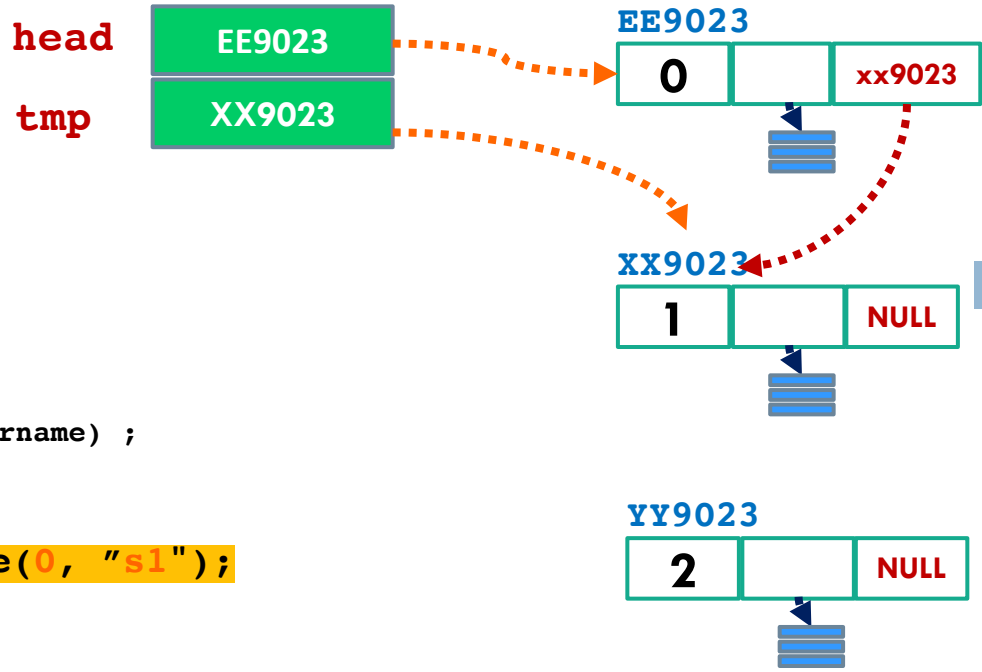
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```



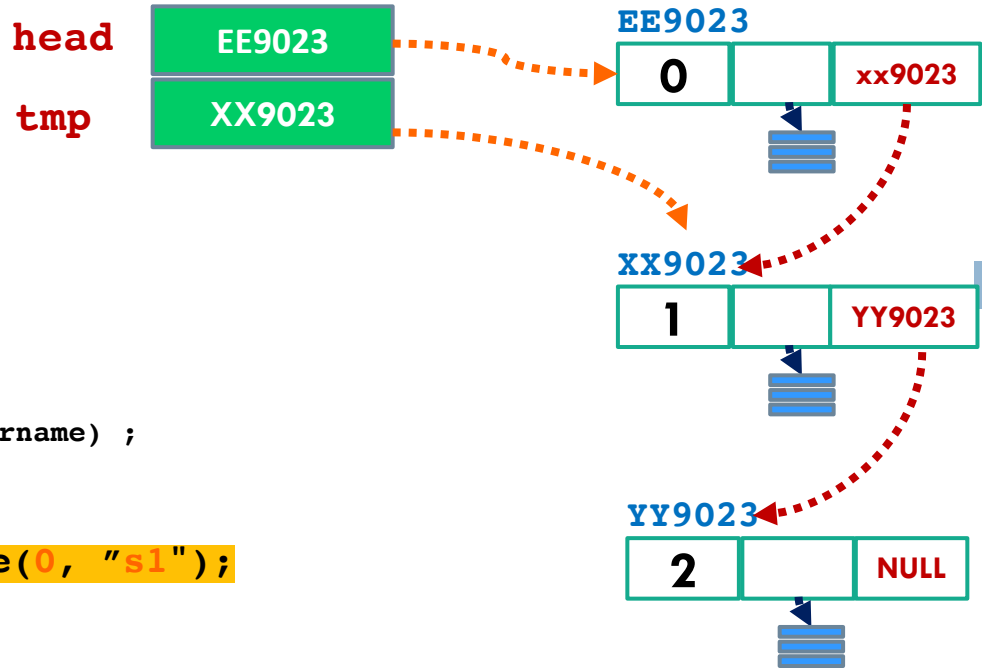
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```



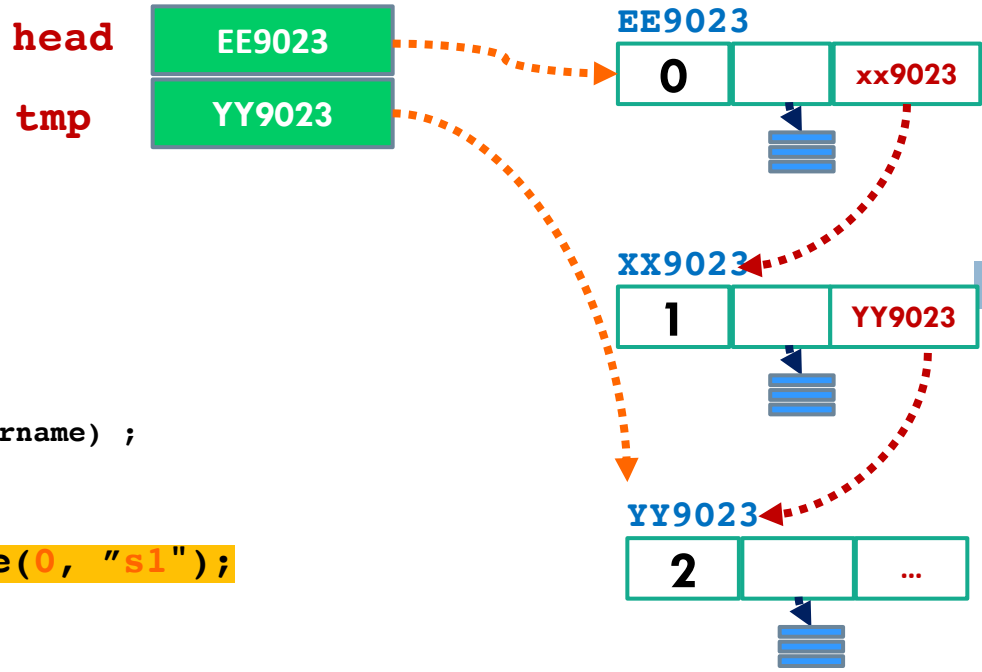
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```



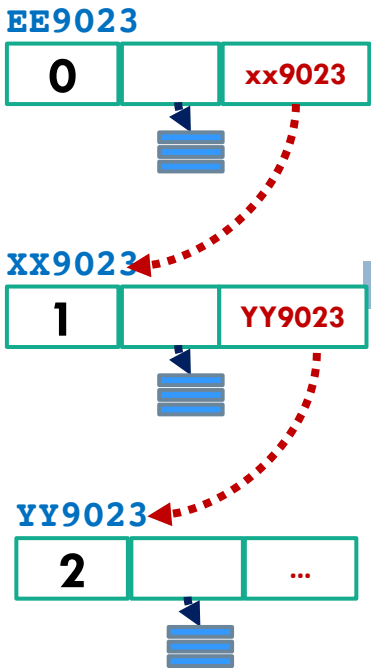
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     return 0;
}
STUDENT * create_student_node(int AM, char *surname) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->AM = AM;
          node->surname = (char *) calloc( strlen(surname) +1,  sizeof(char) );
          strcpy(node->surname, surname);
          node->next = NULL;
          return node;
        }
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
void printlist(STUDENT *nd);
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     printlist(head);
return 0;
}

void printlist(STUDENT *nd){
     STUDENT *  iterator;
     for(iterator=nd; iterator!=NULL; iterator = iterator->next){
               printf("student id: %d \n",  iterator->AM);
     }
}
```

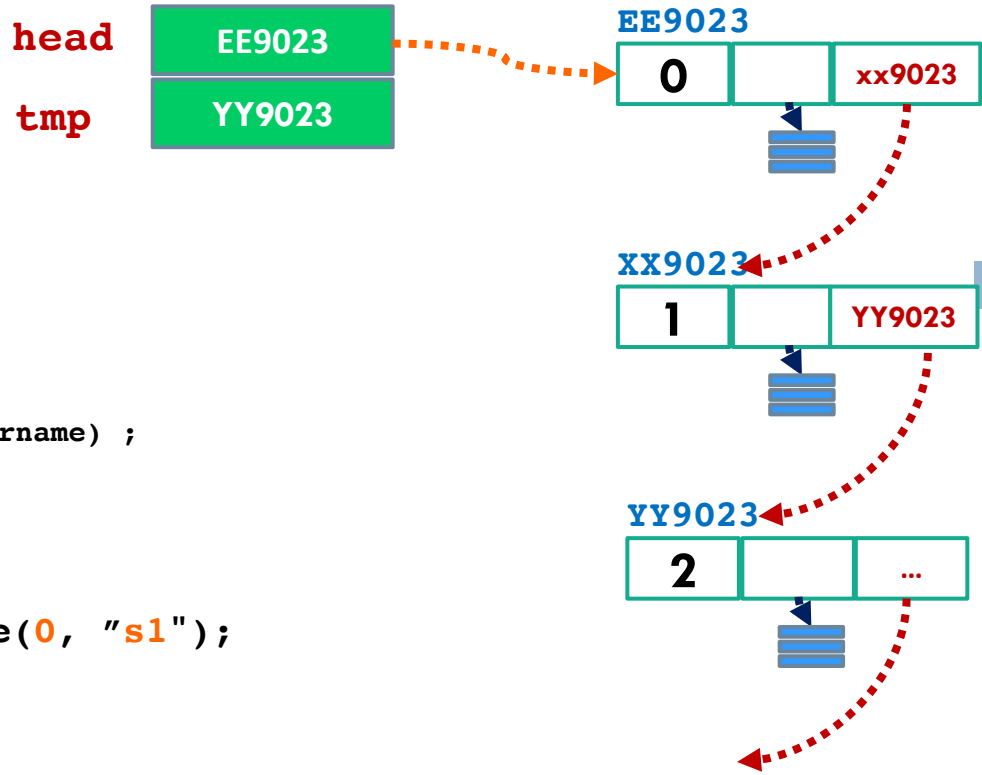
head → EE9023

tmp → YY9023



Εκτύπωση των κόμβων της λίστας.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
   int AM;
   char *surname;
   struct node *next;
}  STUDENT;

STUDENT * create_student_node(int AM, char *surname) ;
void printlist(STUDENT *nd);
STUDENT *  getStudent(STUDENT *nd, int AM);
int main()
{
     STUDENT *head=create_student_node(0, "s1");
     STUDENT *tmp=head;
     int i;
     for(i=1;i<100;i++){
          tmp->next=create_student_node(i, "s");
          tmp=tmp->next;
     }
     if(getStudent(head,50)) printf(("student id: %d \n",  iterator->AM) ;
return 0;
}

STUDENT *  getStudent(STUDENT *nd, int AM){
     STUDENT *  iterator;
     for(iterator=nd; iterator!=NULL; iterator = iterator->next){
               if (iterator->AM==AM) return iterator;
     }
     return NULL;
}
```

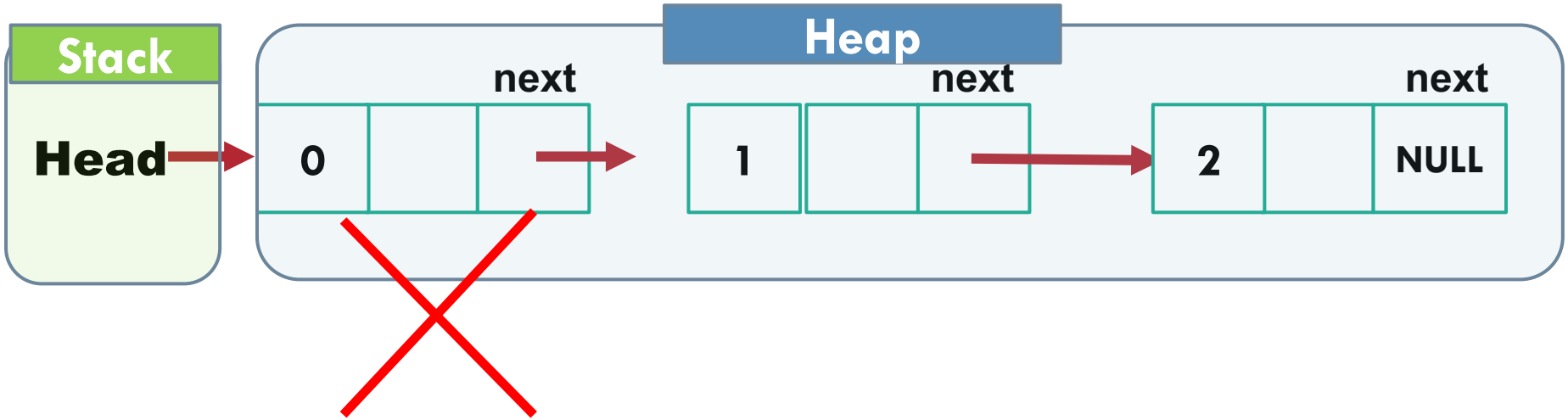


Αναζήτηση κόμβου στη λίστα.

Διαγραφή ενός Κόμβου από μια Λίστα

```
typedef struct node {
                int AM;
                char *surname;
                struct node *next;
} STUDENT;
```

- Τι πρέπει να κάνω αν θέλω να διαγράψω τον πρώτο κόμβο;



Διαγραφή ενός Κόμβου από μια Λίστα

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(head,0);
return 0;
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

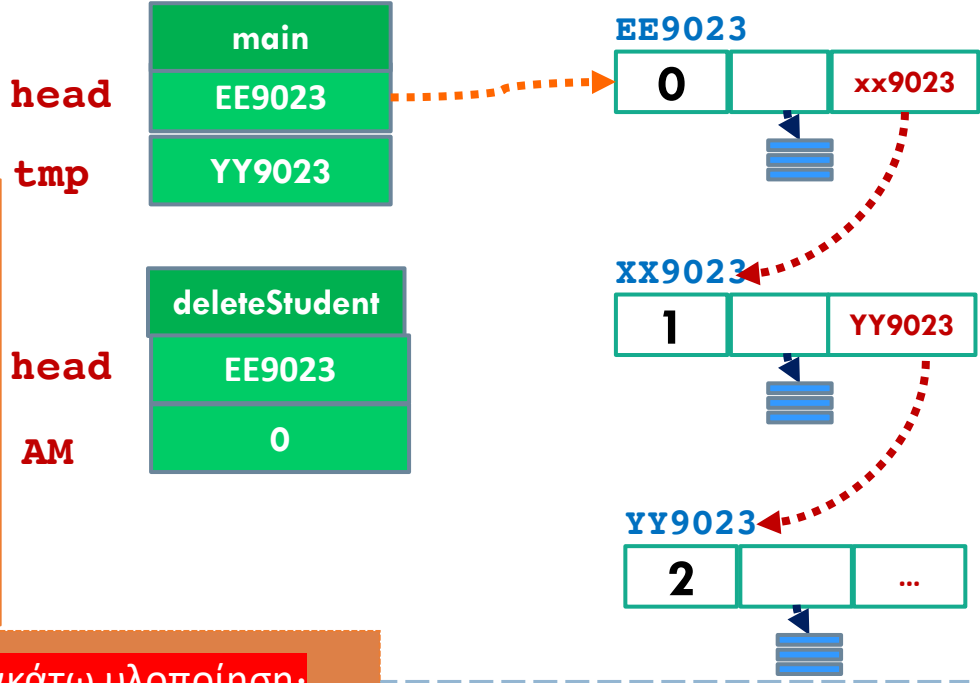
Είναι σωστή η παρακάτω υλοποίηση;

```
void deleteStudent(STUDENT *head, int AM){
    STUDENT *student=NULL;
    student=getStudent(head, AM);
    if(head==student) head=student->next;
    else{
        STUDENT *iterator=head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(head,0);
return 0;
}
```



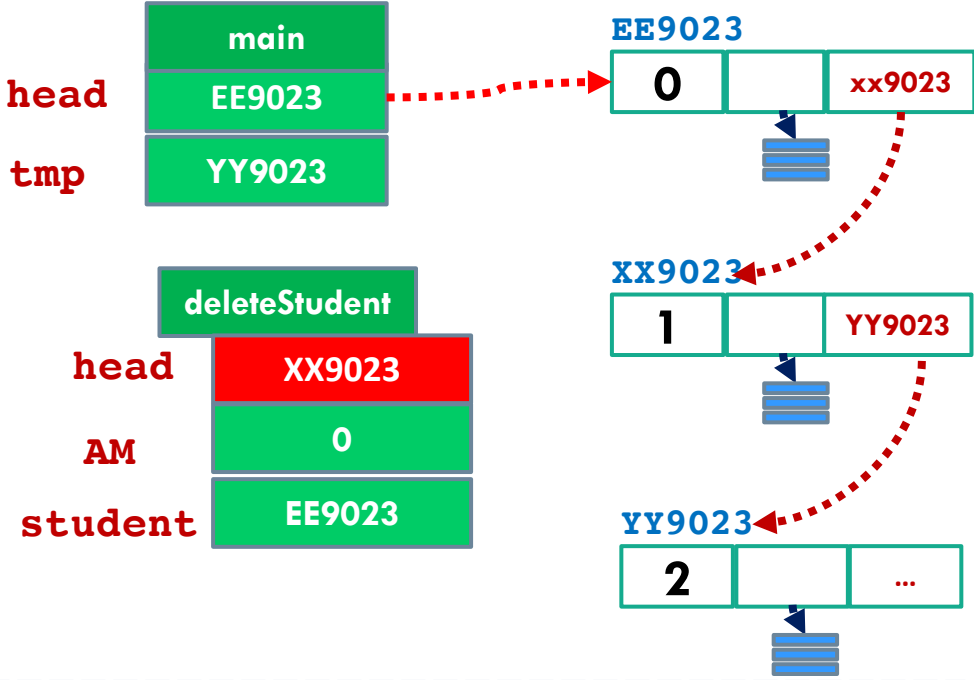
Είναι σωστή η παρακάτω υλοποίηση;

```
void deleteStudent(STUDENT *head, int AM){
    STUDENT *student=NULL;
    student=getStudent(head, AM);
    if(head==student) head=student->next;
    else{
        STUDENT *iterator=head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(head,0);
return 0;
}
```

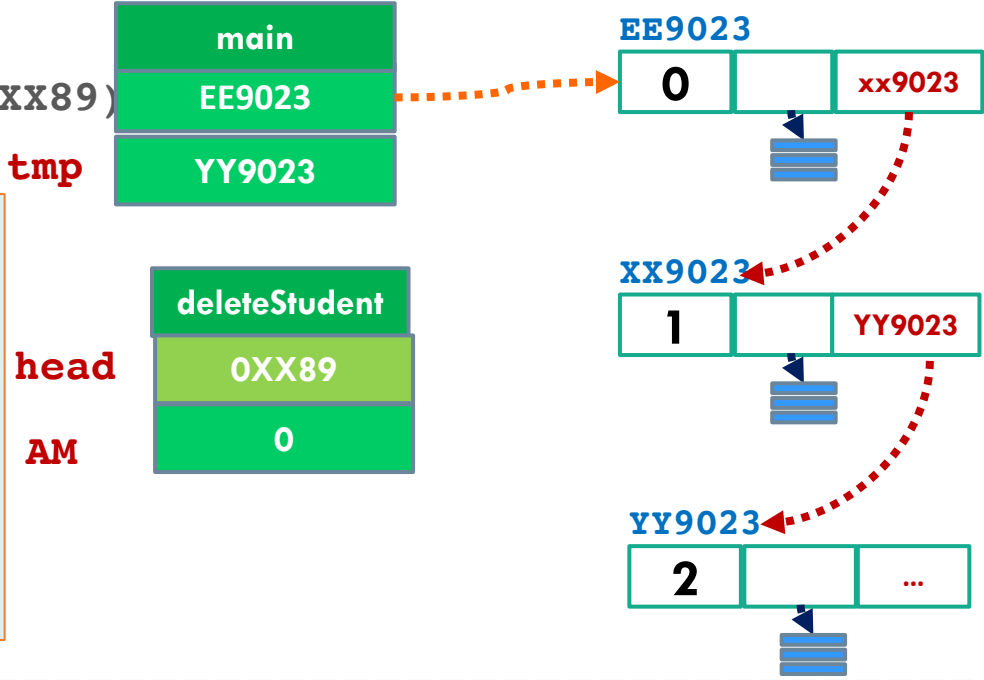



```
void deleteStudent(STUDENT *head, int AM){
    STUDENT *student=NULL;
    student=getStudent(head, AM);
    if(head==student) head=student->next;
    else{
        STUDENT *iterator=head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,0);
return 0;
}
```

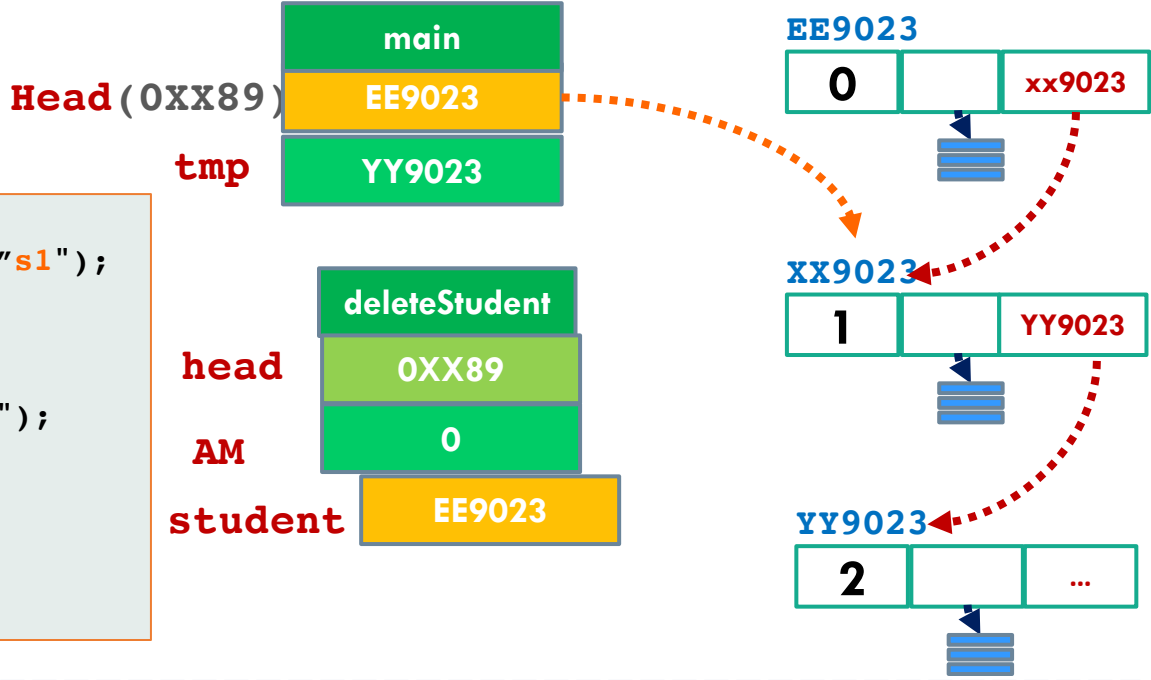



```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iterator=*head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,0);
return 0;
}
```

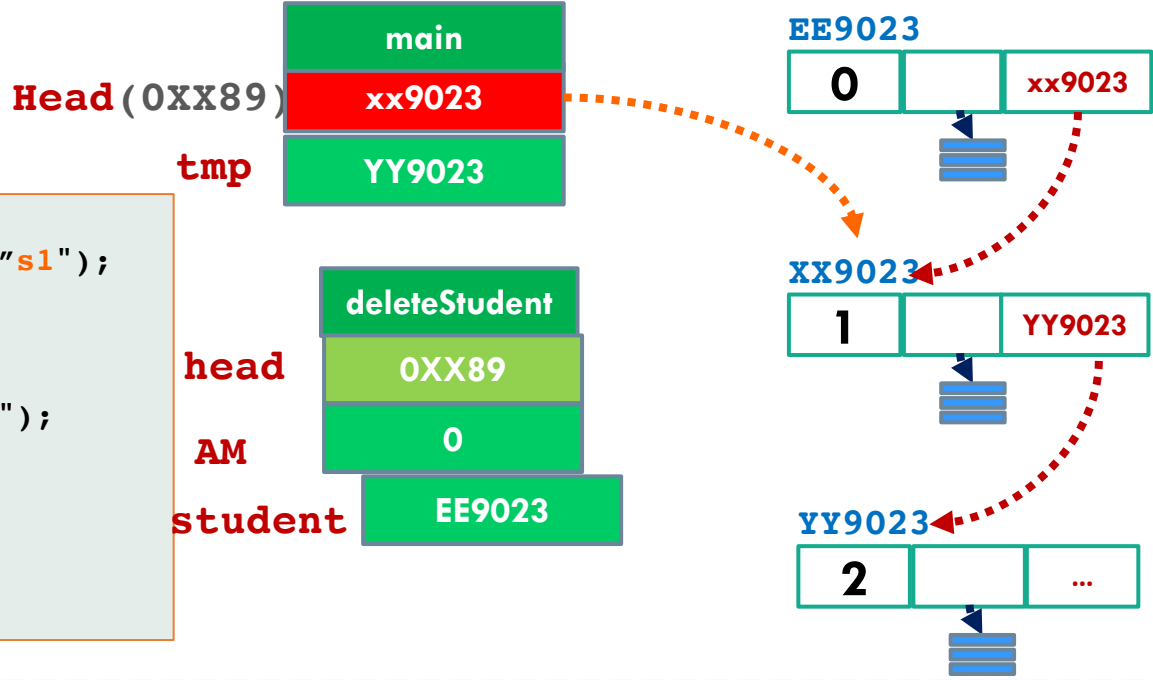



```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student)  head=student->next;
    else{
        STUDENT *iterator=*head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,0);
return 0;
}
```

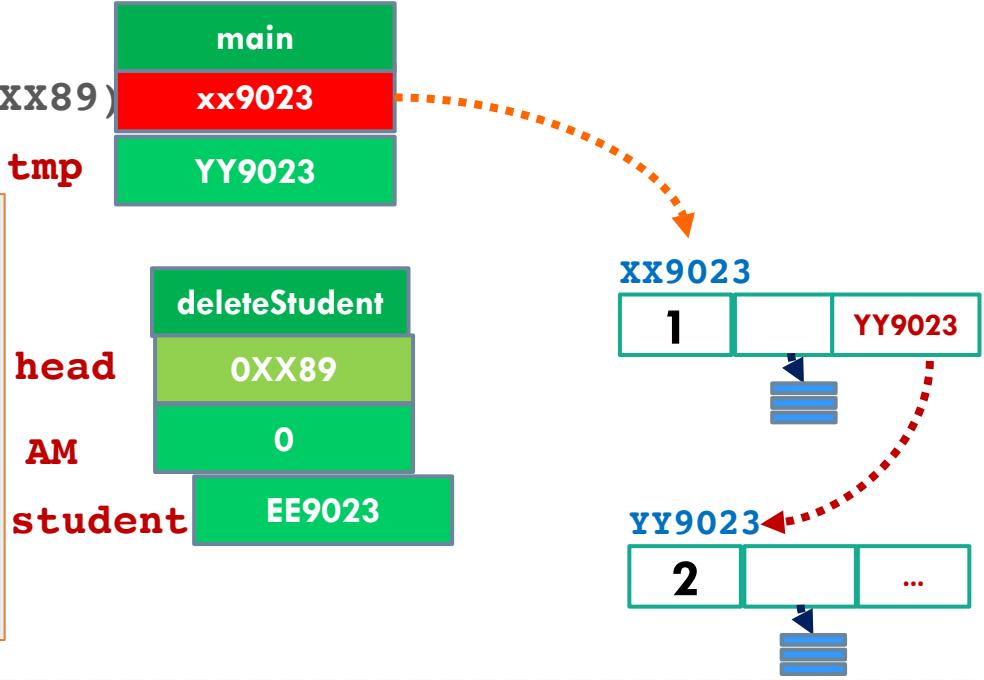



```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iterator=*head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iterator=*head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

9η Εβδομάδα: Διασυνδεδεμένες Λίστες

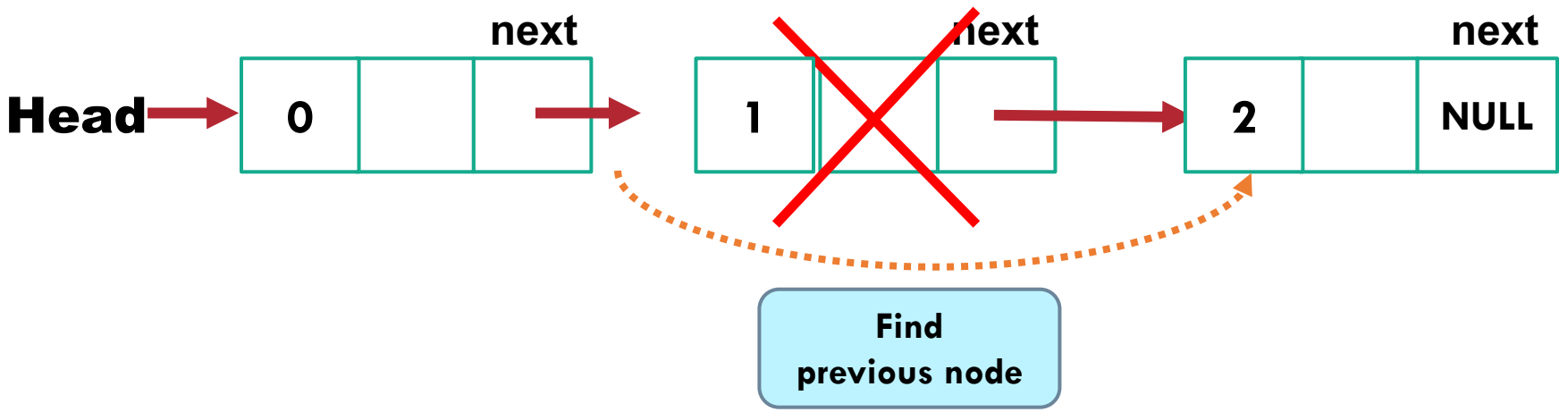
Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Διαγραφή ενός Κόμβου από μια Λίστα

```
typedef struct node {
                int AM;
                char *surname;
                struct node *next;
} STUDENT;
```

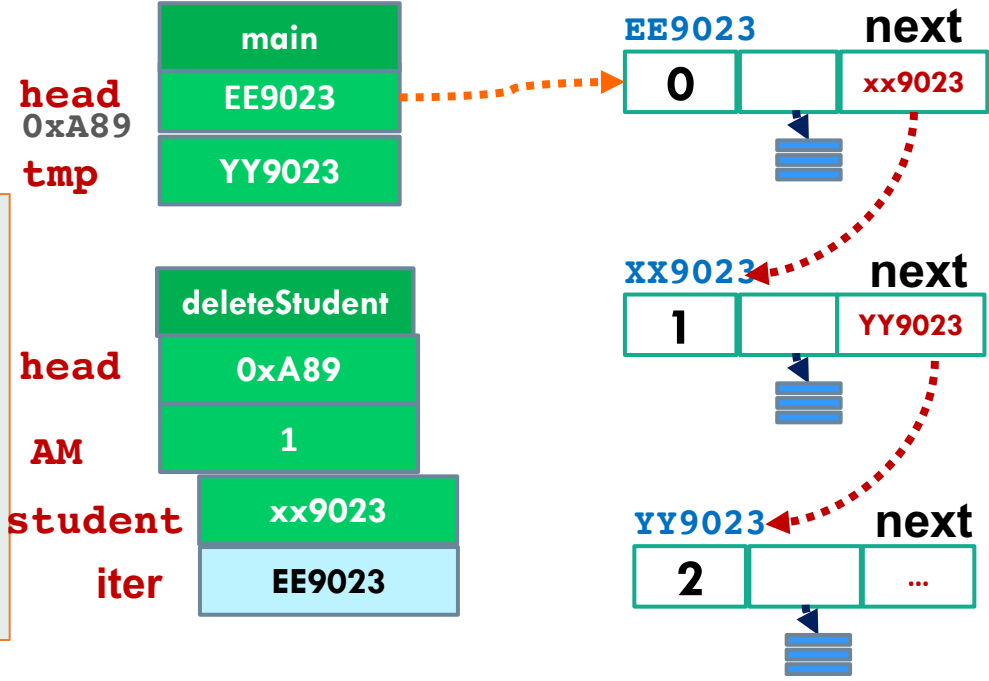
- Τι πρέπει να κάνω αν θέλω να διαγράψω έναν ενδιάμεσο κόμβο;



Διαγραφή ενός Κόμβου μια

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=1

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,1);
return 0;
}
```



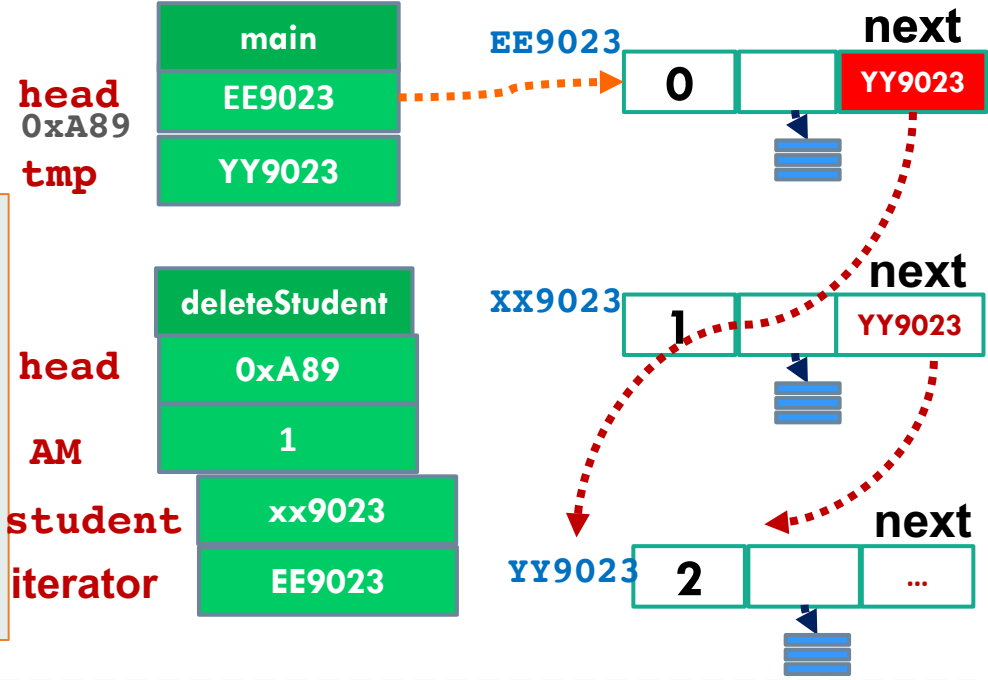

```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iter=*head;
        for(; iter != NULL && iter >next != student ; iter = iter >next);
        if(iter!=NULL) iter >next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Find previous node

Διαγραφή ενός Κόμβου

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=1

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,1);
return 0;
}
```



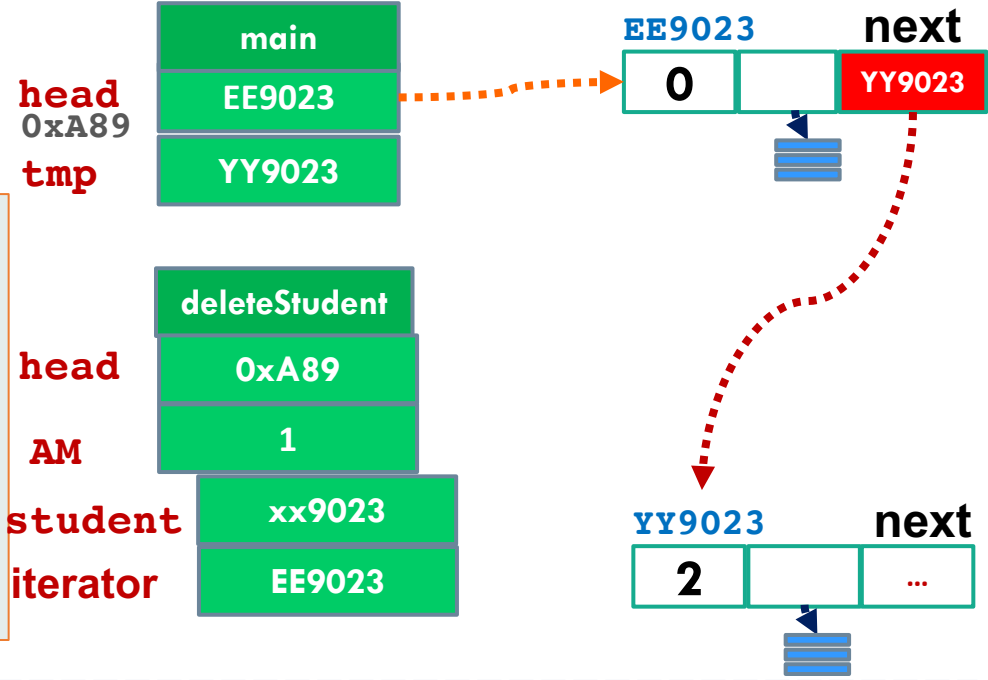

```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iterator=*head;
        for(; iterator != NULL && iterator->next != student ;
                iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Find previous node

Διαγραφή ενός Κόμβου μια

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,1);
return 0;
}
```



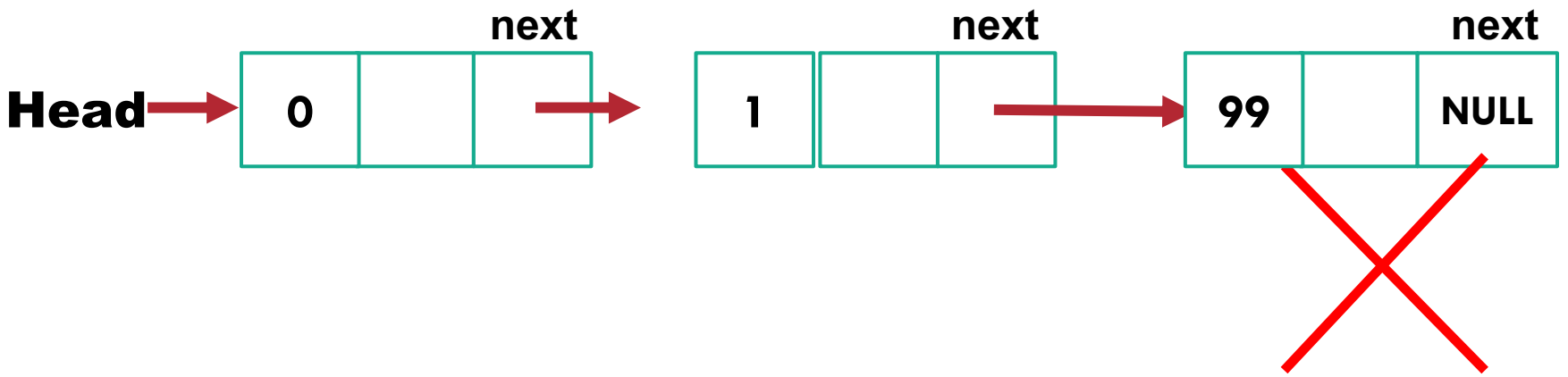

```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) head=student->next;
    else{
        STUDENT *iterator=*head;
        for(; iterator != NULL && iterator->next != student ;
              iterator =iterator->next);
        if(iterator!=NULL) iterator->next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Find previous node

Διαγραφή ενός Κόμβου από μια Λίστα

```
typedef struct node {
                int AM;
                char *surname;
                struct node *next;
} STUDENT;
```

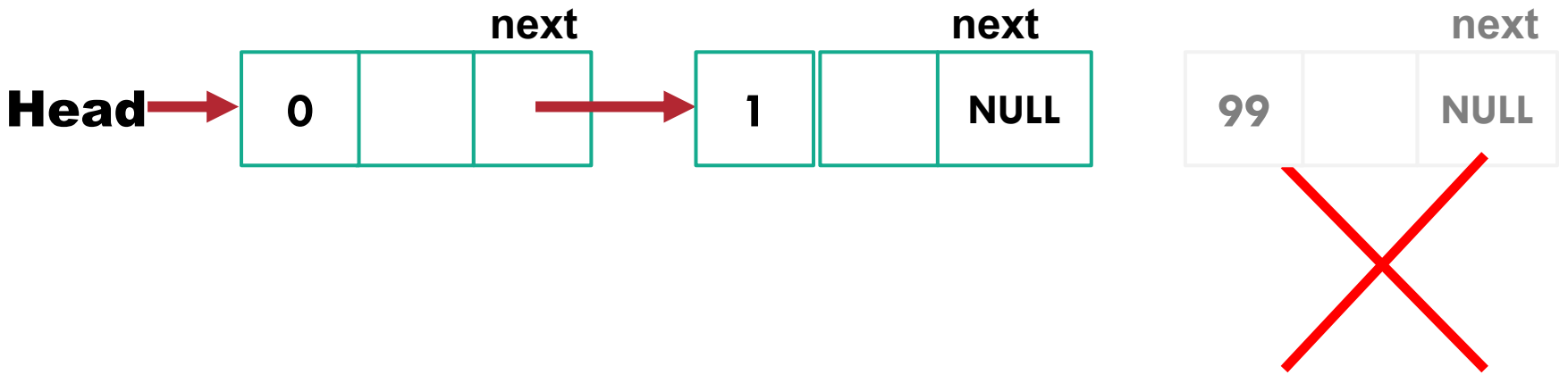
- Τι πρέπει να κάνω αν θέλω να διαγράψω τον τελευταίο κόμβο;



Διαγραφή ενός Κόμβου από μια Λίστα

```
typedef struct node {
                int AM;
                char *surname;
                struct node *next;
} STUDENT;
```

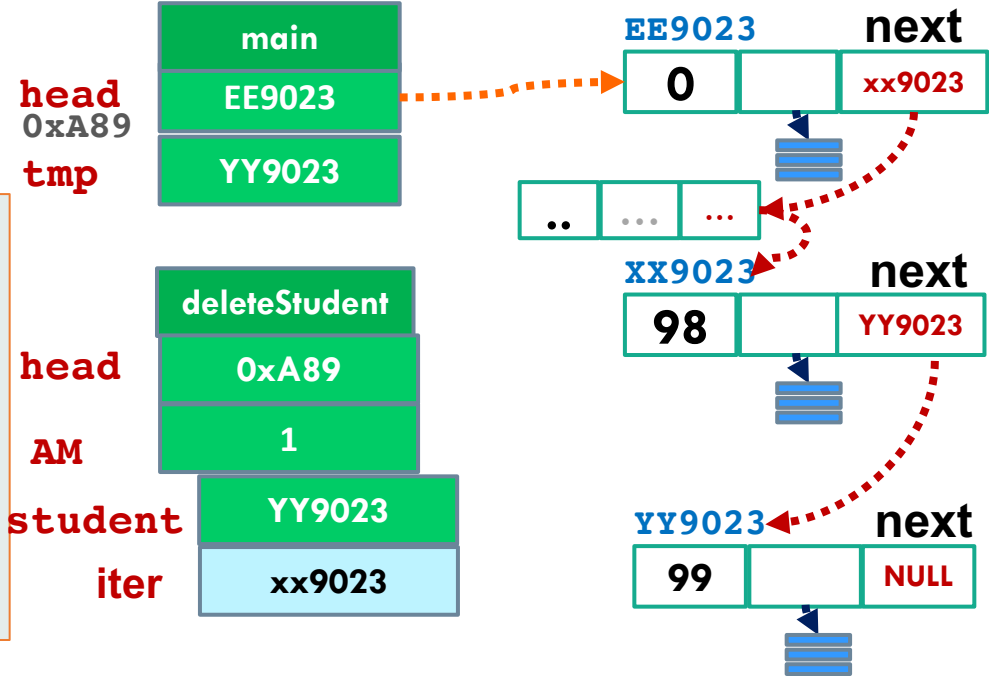
- Τι πρέπει να κάνω αν θέλω να διαγράψω τον τελευταίο κόμβο;



Διαγραφή ενός Κόμβου μια

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=99

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,99);
return 0;
}
```



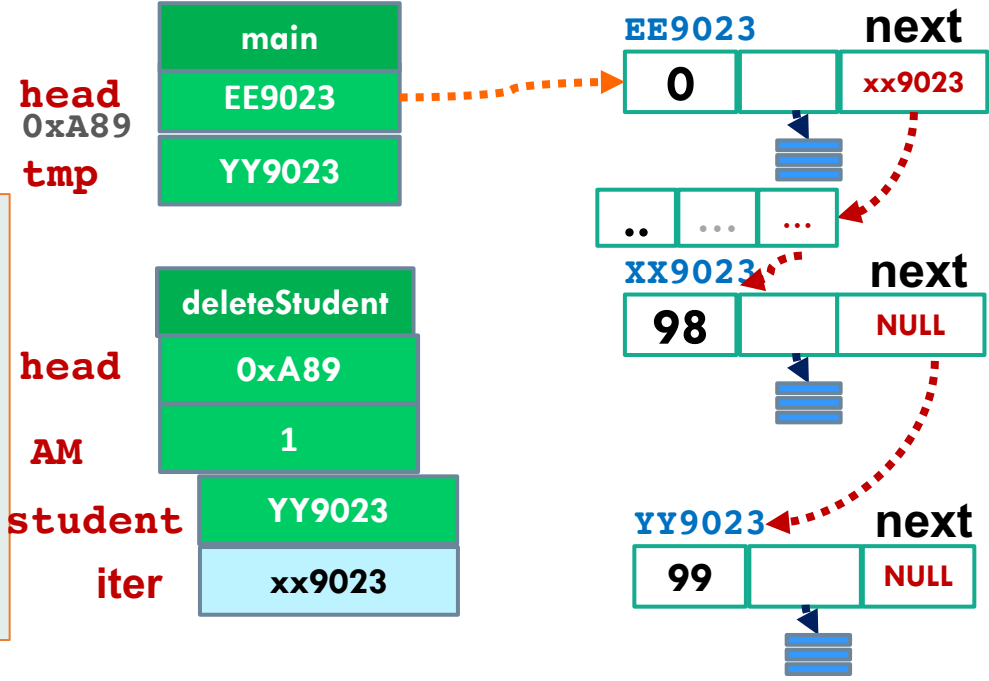

```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iter=*head;
        for(; iter != NULL && iter >next != student ; iter = iter >next);
        if(iter!=NULL) iter >next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Find previous node

Διαγραφή ενός Κόμβου μια

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=99

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,99);
return 0;
}
```



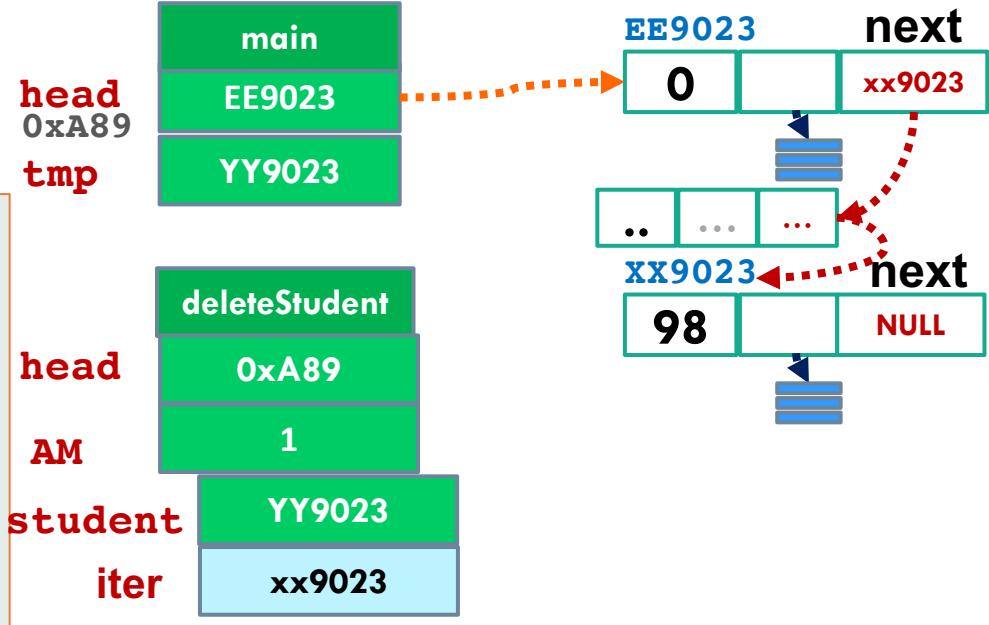

```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iter=*head;
        for(; iter != NULL && iter >next != student ; iter = iter >next);
        if(iter!=NULL) iter >next=student->next;
    }
    free(student->surname);
    free(student);
}
```

Find previous node

Διαγραφή ενός Κόμβου

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=99

```
int main(){
STUDENT *head=create_student_node(0, "s1");
STUDENT *tmp=head;
int i;
for(i=1;i<100;i++){
  tmp->next=create_student_node(i, "s");
  tmp=tmp->next;
}
deleteStudent(&head,99);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=getStudent(*head, AM);
    if(*head==student) *head=student->next;
    else{
        STUDENT *iter=*head;
        for(; iter != NULL && iter >next != student ; iter = iter >next);
        if(iter!=NULL) iter >next=student->next;
    }
    free(student->surname);
    free(student);
}
```

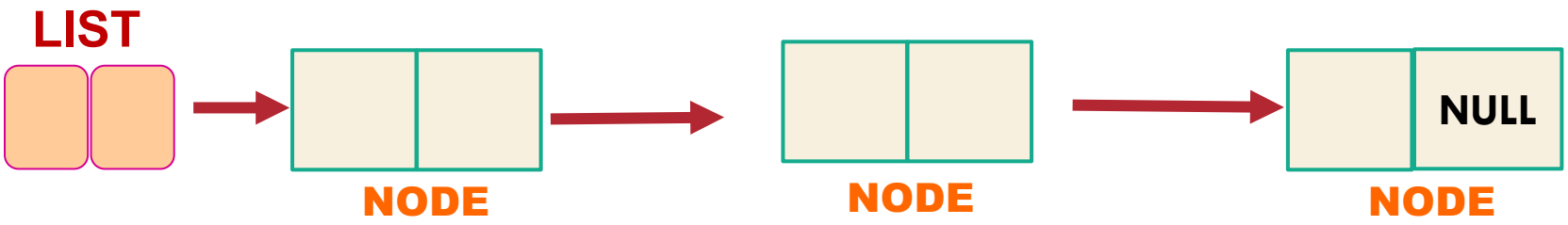
Find previous node

Μια διαφορετική προσέγγιση

```
typedef struct {
    NODE  *head;
    int size;
} LIST;
```

```
typedef struct node {
                int val;
                struct node *next;
} NODE;
```

Μπορούμε μέσω της LIST να αναφερόμαστε στον πρώτο κόμβο της λίστας και να καταγράφουμε λοιπά χαρακτηριστικά της λίστας ανάλογα με το πρόβλημα (π.χ. size, numberof_odd_nodes, number_of_even_nodes, κτλ):



```c
#include <stdio.h>
#include <stdlib.h>
typedef struct node {
        int         val;
        struct node *next;
} NODE;

typedef struct {
     NODE  *head;
     int size;
} LIST;

void createLinkedList(LIST *,int );
void printlist(LIST *);

int main()
{
    LIST mylist={NULL,0};
    createLinkedList(&mylist,10);
    printlist(&mylist);
    return 0;
}
```

```c
void createLinkedList(LIST * list,int x){
    NODE *tmp;
    int i=0;
    for(;i<x;i++){
          if(list->head==NULL){
             list->head=createNode(list, i);
             tmp=list->head;
          }
          else{
              tmp->next=createNode(list, i);
              tmp=tmp->next;
          }
    }
    return ;
}
```

```c
NODE* createNode (LIST * list, int value) {
        NODE *node;
        node = (NODE *)malloc(sizeof(NODE));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          list->size=list->size+1;
          node->val = value;
          node->next = NULL;
          return node;
        }
}
```

Αναζήτηση Κόμβου σε Ευθύγραμμες Απλά Συνδεδεμένες Λίστες

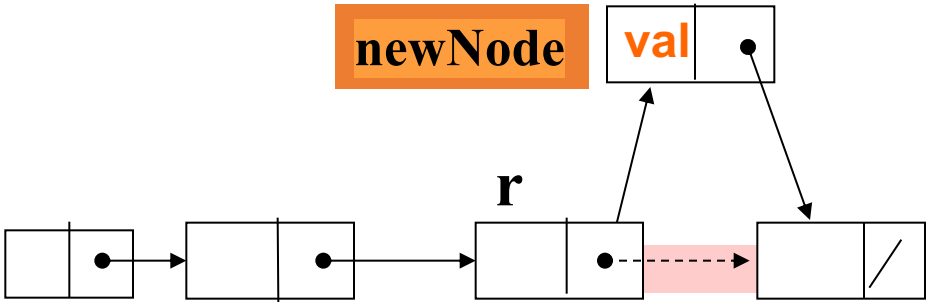
Με παρόμοιο τρόπο μπορούμε να ορίσουμε διαδικασία findpointer(LIST *L, key val) που επιστρέφει δείκτη προς κόμβο της λίστας που περιέχει το στοιχείο val, αν υπάρχει.

```
NODE *findpointer(LIST *list, int value) {
  NODE *p = list->head;
  while (p != NULL) {
    if (p->val == value) {
          return p;
    }
    p = p->next;
  }
  return NULL;
}
```

Ευθύγραμμες Απλά Συνδεδεμένες Λίστες

Εισαγωγή Νέου Κόμβου μετά από συγκεκριμένο κόμβο "r"

```
int insert(LIST *L, NODE *r, int val) {
        NODE * newNode=createNode(L, val);
        if(newNode==NULL) return 1;
        newNode->next = r->next;
        r->next =newNode;
       return 0;
}
```



Εισαγωγή Νέου Κόμβου μετά από τον κόμβο "0"


```
int insert(LIST *L, NODE *r, int val);
int main()
{
   LIST mylist={NULL,0};
   createLinkedList(&mylist,10);
   printlist(&mylist);

   insert(&mylist,  findpointer(&mylist,0), 100);
   printlist(&mylist);
   return 0;
}
```

```
List size:10
student id: 0  66405910  next:66405920
student id: 1  66405920  next:66405930
student id: 2  66405930  next:66405940
student id: 3  66405940  next:66405950
student id: 4  66405950  next:66405960
student id: 5  66405960  next:66405970
student id: 6  66405970  next:66405980
student id: 7  66405980  next:66405990
student id: 8  66405990  next:664059A0
student id: 9  664059A0  next:0
List size:11
student id: 0  66405910  next:664059B0
student id: 100  664059B0  next:66405920
student id: 1  66405920  next:66405930
student id: 2  66405930  next:66405940
student id: 3  66405940  next:66405950
student id: 4  66405950  next:66405960
student id: 5  66405960  next:66405970
student id: 6  66405970  next:66405980
student id: 7  66405980  next:66405990
student id: 8  66405990  next:664059A0
student id: 9  664059A0  next:0
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/