LINKED LISTS

A linked list is comprised of a series of nodes, each node containing a data element, and a pointer to the next node, e.g.,

```
List ───► Info ───► Next ───► info ───► Next ───► info ───► Next ───► info ───► Null
```

A structure which contains a data element and a pointer to the next node is created by,

```
struct list {
    int    value;
    struct list *next;
};
```

This defines a new data structure called list (actually the definition of a node), which contains two members. The first is an integer called value. The second is called next, which is a pointer to another list structure (or node). Suppose that we declare two structures to be of the same type as list, e.g.,

```
struct list n1, n2;
```

The next pointer of structure n1 may be set to point to the n2 structure by

```
/* assign address of first element in n2 to the pointer next of the n1 structure */
```

```
n1.next = &n2;
```

which creates a link between the two structures.

```
/* LLIST.C  Program to illustrate linked lists */
#include <stdio.h>

struct list {
    int    value;
    struct list *next;
};
main()
{
    struct list n1, n2, n3;
    int    i;
    n1.value = 100;
    n2.value = 200;
    n3.value = 300;
    n1.next = &n2;
    n2.next = &n3;
    i = n1.next->value;
    printf("%d\n", n2.next->value);
}
```
Not only this, but consider the following

```c
n1.next = n2.next;  /* removes n2 from the list */
n2_3.next = n2.next; /* adds struct n2_3 */
n2.next = &n2_3;
```

In using linked list structures, it is common to assign the value of 0 to the last pointer in the list, to indicate that there are no more nodes in the list, e.g.,

```c
n3.next = 0;
```

**Traversing a linked list**

This program uses a pointer called `list_pointer` to cycle through the linked list.

```c
/* Program to illustrate traversing a list */
#include <stdio.h>
struct list {  
    int value;
    struct list *next;
};
main()
{
    struct list n1, n2, n3, n4;
    struct list *list_pointer = &n1;
    n1.value = 100;
    n1.next = &n2;
    n2.value = 200;
    n2.next = &n3;
    n3.value = 300;
    n3.next = &n4;
    n4.value = 400;
    n4.next = 0;
    while( list_pointer != 0 ) {  
        printf("%d\n", list_pointer->value);
        list_pointer = list_pointer->next;
    }
}
```

**Lab Work**

Compile and run the following program and understand how it works. Enter some data and see the results.

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define RECORDS 6

struct student{
    char name[25];
    long id_num;
    float GPA;
    struct student *next;
};
```
student* getnode(void);
void freenode(student*);
student* addstudent(student *,char [],long ,float );
void printlist(student *);
void freelist(student *);
void main()
{
    char name[25];
    float GPA;
    long id_num;
    student *start;
    start=NULL;
    for (int i=0;i<RECORDS;i++)
    {
        printf("Enter Students name ");
        scanf("%s",&name);
        printf("\n Enter Students id number ");
        scanf("%ld",&id_num);
        printf("\n Enter Students GPA ");
        scanf("%f",&GPA);
        start=addstudent(start,name,id_num,GPA);
    }
    printlist(start);
    freelist(start);
}
student* getnode(void)
{
    student *p;
p=(student*)malloc(sizeof(student));
    return p;
}
student *addstudent(student *p,char name[],long id_num,float GPA){
    student *newElm,Elm;
    newElm=getnode();
    strcpy(newElm->name,name);
    newElm->id_num=id_num;
    newElm->GPA=GPA;
    newElm->next=NULL;
    if(p==NULL)
        return newElm;
    else{
        Elm=p;
        while(Elm->next!=NULL)
            Elm=Elm->next;
        Elm->next=newElm;
        return p;
    }
}
void printlist(student *p){
    3
    3
student *q;  
q = getnode();  
q = p;  
do {  
    printf("%s\t", q->name);  
    printf("%ld\t", q->id_num);  
    printf("%fn", q->GPA);  
    q = q->next;  
} while(q!=NULL);  
}  
void freelist(student *p){  
student *q,*s;  
q=p;  
do {  
    s = q->next;  
    free(q);  
    q = s;  
} while (s!=NULL);  
}

**Homework 6 (due one week):**

Write a C program for a library automation which gets the ISBN number, name, author and publication year of the books in the library. The status will be filled by the program as follows: if publication year before 1985 the *status is reference* else *status is available*. The information about the books should be stored inside a linked list. The program should have a menu and the user inserts, displays, and deletes the elements from the menu by selecting options. The following data structure should be used.

```c
struct list{  
    char ISBN[20];  
    char NAME[20];  
    char AUTHOR[20];  
    int YEAR;  
    char STATUS[20];  
    struct list *next;  
}INFO;
```

The following menu should be used in the program.

- Press 1. to insert a book
- Press 2. to display the book list
- Press 3. to delete a book from list

Hint: use `strcpy` to fill STATUS.

**NOTES:**

1) Please Send your Homework in the following Emails, but remember who is your lab instructor

**emu.clab2@gmail.com** for Pouya's Student
2) Subject of email Should include student Number + Homework Number
    For Example: "St. 1500012 Homework #1"

3) Your homework should be saved with your student number and attached as notepad.
    For Example 1500012.txt