

CS11001/CS11002

**Programming and Data
Structures (PDS)**

(Theory: 3-1-0)

The List ADT

- Let us now define a new ADT which has not been encountered earlier in your math courses.
- We call this ADT the ordered list.
 - It is a list of elements, say characters, in which elements are ordered, i.e., there is a zeroth element, a first element, a second element, and so on, and in which repetitions of elements are allowed.

Functions on the List ADT

- `L = init();`
 - Initialize L to an empty list.
- `L = insert(L,ch,pos);`
 - Insert the character `ch` at position `pos` in the list L and return the modified list. Report error if `pos` is not a valid position in L.
- `delete(L,pos);`
 - Delete the character at position `pos` in the list L. Report error if `pos` is not a valid position in L.
- `isPresent(L,ch);`
 - Check if the character `ch` is present in the list L. If no match is found, return -1, else return the index of the leftmost match.
- `getElement(L,pos);`
 - Return the character at position `pos` in the list L. Report error if `pos` is not a valid position in L.
- `print(L);`
 - Print the list elements from start to end.

Some functions on the List ADT

```
#include<stdio.h>
#define MAXLEN 100

typedef struct {
    int len;
    char element[MAXLEN];
} olist;

olist init()
{
    olist L;
    L.len = 0;
    return L;
}

void print(olist L)
{
    int i;

    for(i = 0; i < L.len; ++i) printf("%c", L.element[i]);
}

olist insert(olist L , char ch , int pos)
{
    int i;

    if ((pos < 0) || (pos > L.len)) {
        fprintf(stderr, "insert: Invalid index %d\n",
            pos);
        return L;
    }
    if (L.len == MAXLEN) {
        fprintf(stderr, "insert: List already full\n");
        return L;
    }
    for (i = L.len; i > pos; --i) L.element[i] =
        L.element[i-1];
    L.element[pos] = ch;
    ++L.len;
    return L;
}
```

The main function

```
main()
{
    olist L;
    L=init();
    L=insert(L,'a',0);
    print(L);
    printf("\n");
    L=insert(L,'b',0);
    print(L);
    printf("\n");
    L=insert(L,'c',2);
    print(L);
    printf("\n");
}
```

- a
- ba
- bac

The delete procedure

```
olist delete(olist L, int pos)
{
    int i;

    if ((pos < 0) || (pos >= L.len)) {
        fprintf(stderr, "delete: Invalid index %d\n", pos);
        return L;
    }
    for (i = pos; i <= L.len - 2; ++i) L.element[i] = L.element[i+1];
    --L.len;
    return L;
}
```

Lists with Linked Lists (using pointers)

The definition

```
#include<stdio.h>
#include<malloc.h>

typedef struct list{
    char value;
    struct list *next;
}node;

typedef node* olist;
```

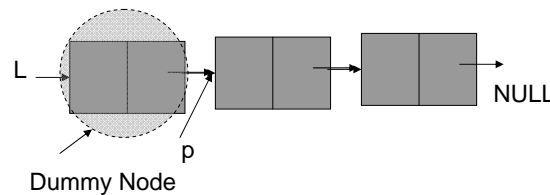
The init function

```
olist init()
{
    olist L;
    L=(olist)malloc(sizeof(node));
    L->value='\0';
    L->next=NULL;
    return(L);
}
```



The print function

```
void print(olist L)
{
    node *p;
    p=L->next;
    while(p!=NULL)
    {
        printf("%c",p->value);
        p=p->next;
    }
    printf("\n");
}
```



The insert function

```
olist insert(olist L, char ch, int pos)
{
    int i;
    node *p;
    node *t;

    if(pos<0)
    {
        fprintf(stderr,"insert: Error in
        index\n");
        return(L);
    }
    p=L;
    i=0;
    while(i<pos)
```

```
{
    p=p->next;
    if(p==NULL){
        fprintf(stderr,"insert:
        invalid index\n");
        return(L);
    }
    i++;
}
t=(node
*)malloc(sizeof(node));
t->value=ch;
t->next=p->next;
p->next=t;
return(L);
}
```

Inserting into an empty list (pos=0)

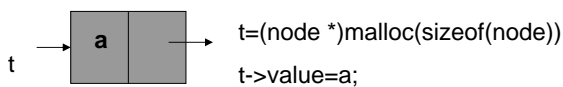
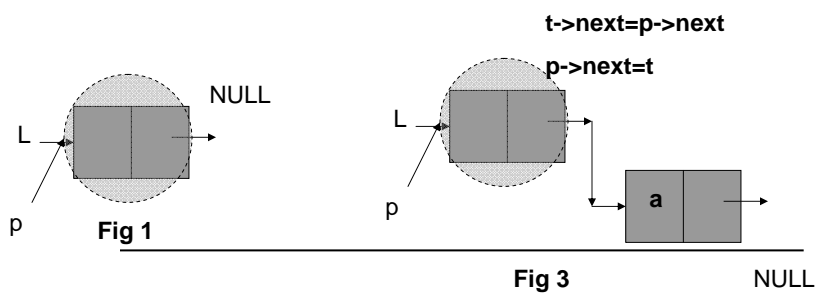
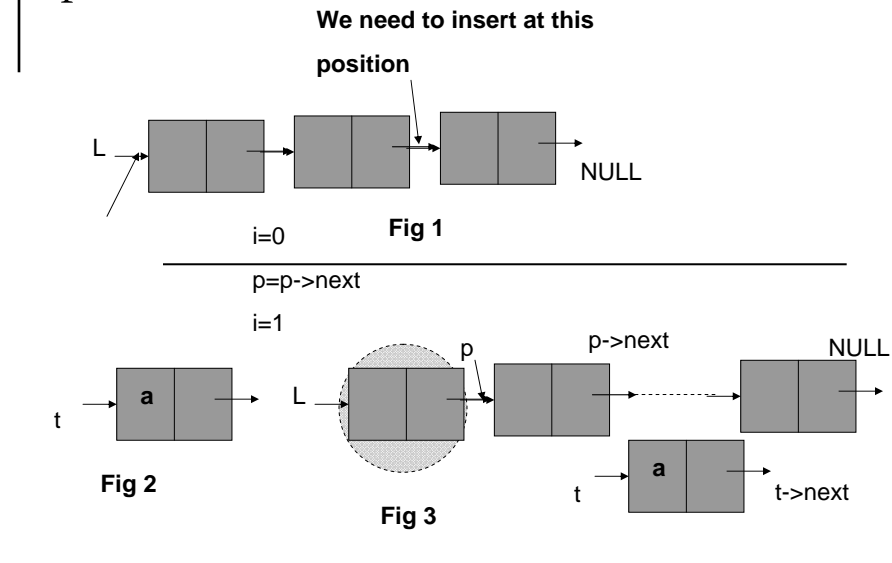


Fig 2

```
t=(node *)malloc(sizeof(node))
t->value=a;
```

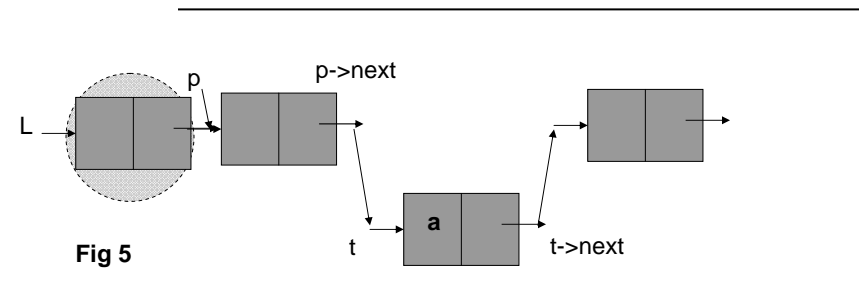
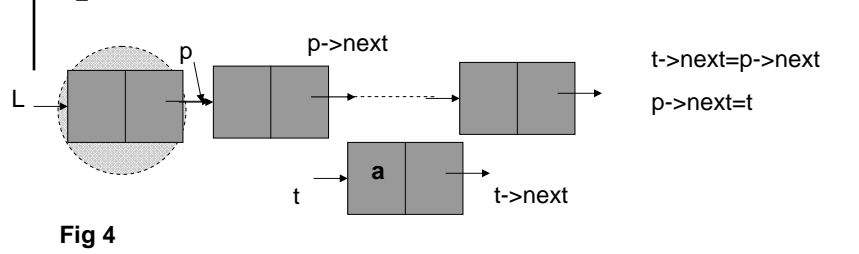
Inserting into an existing list with 2 nodes

at pos=1



Inserting into an existing list with 2 nodes

at pos=1



The delete function

```
olist delete(olist L, int pos)
{
    int i=0;
    node *p;
    if(pos<0){
        fprintf(stderr,"delete: invalid
index\n");
        return(L);
    }
    p=L;
    i=0;
```

```
while(i<pos)
{
    p=p->next;
    if(p->next==NULL)
    {
        fprintf(stderr,"delete: invalid
index\n");
        return(L);
    }
    i++;
}
p->next=p->next->next;
return(L);
}
```

The main function

```
main()
{
    olist L;
    L=init();
    print(L);
    insert(L,'a',0);
    print(L);
    insert(L,'b',1);
    print(L);
    insert(L,'c',1);
    print(L);
    delete(L,3);
    print(L);
    delete(L,2);
    print(L);
    delete(L,0);
    print(L);
}
```


- a
- ab
- acb
- delete: invalid index
- acb
- ac
- c