

Topic 5

Nodes and Linked Structures

CSE1303 Part A

Data Structures and Algorithms

Overview of Topic

- Review List Implementations.
 - Nodes.
 - Linked Stacks.
 - Linked Queues
 - Linked Lists.
 - Other List Operations
- 
- Today's
Lecture

Lists

1

0

3

1

6

2

10

3

15

4

A List ADT

A sequence of elements together with these operations:

- Initialize the list.
- Determine whether the list is empty.
- Determine whether the list is full.
- Find the size of the list.
- Insert an item anywhere in the list.
- Delete an item anywhere in a list.
- Go to a particular position in a list.

Insertion

6

Inserted at position 2

Before:



1

3

10

15

0

1

2

3

After:

1

3

6

10

15

0

1

2

3

4

A List ADT

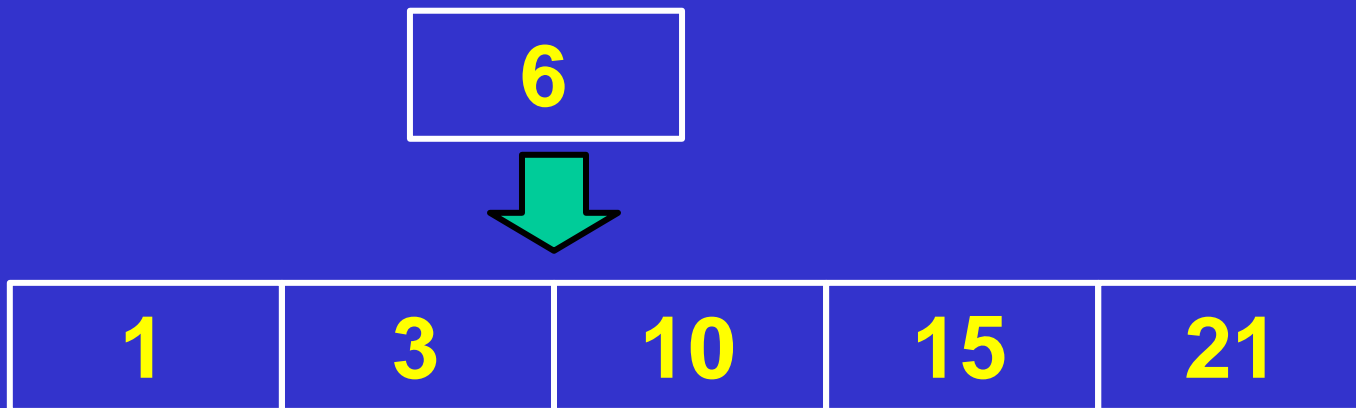
A sequence of elements together with these operations:

- Initialize the list.
- Determine whether the list is empty.
- Find the size of the list.
- Insert an item anywhere in the list.
- Delete an item anywhere in a list.
- Go to a particular position in a list.



List needs to be able to expand

Simple List Implementation



Expansion and Insertion



1	3	6	10	15	21
---	---	---	----	----	----

Disadvantages

- Lots of memory needs to be allocated.
- Lots of copying needs to be done.

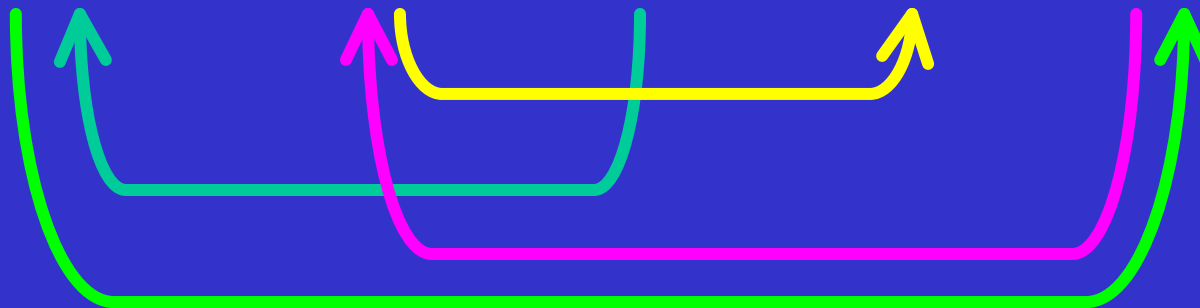
Linked List Implementation: Using Pointers

insert: 6

start



0x2000	0x2008	0x2010	0x2018	0x2020
3	15	1	21	10
0x2020	0x2018	0x2000	NULL	0x2008



Method 1

start

0x30F8

newStart



0x30F0	0x30F8	0x3100	0x3108	0x3110	0x3118
1	3	6	10	15	21
0x30F8	0x3100	0x3108	0x3110	0x3118	NULL

Method 2

start



0x2000

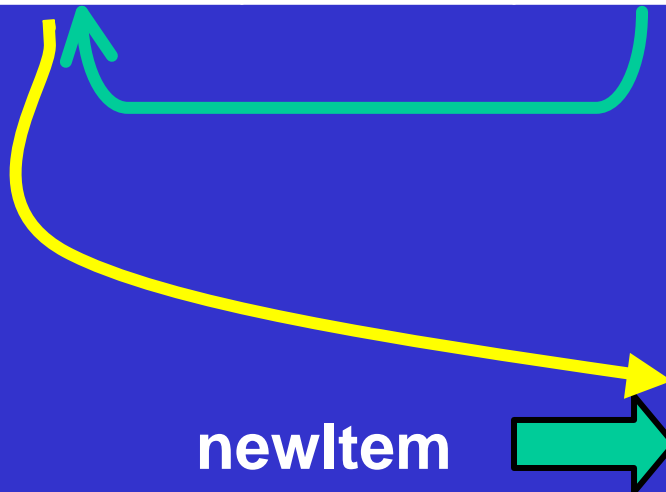
0x2008

0x2010

0x2018

0x2020

3	15	1	21	10
0x2020	0x2018	0x2000	NULL	0x2008



0x30F0

6

newItem

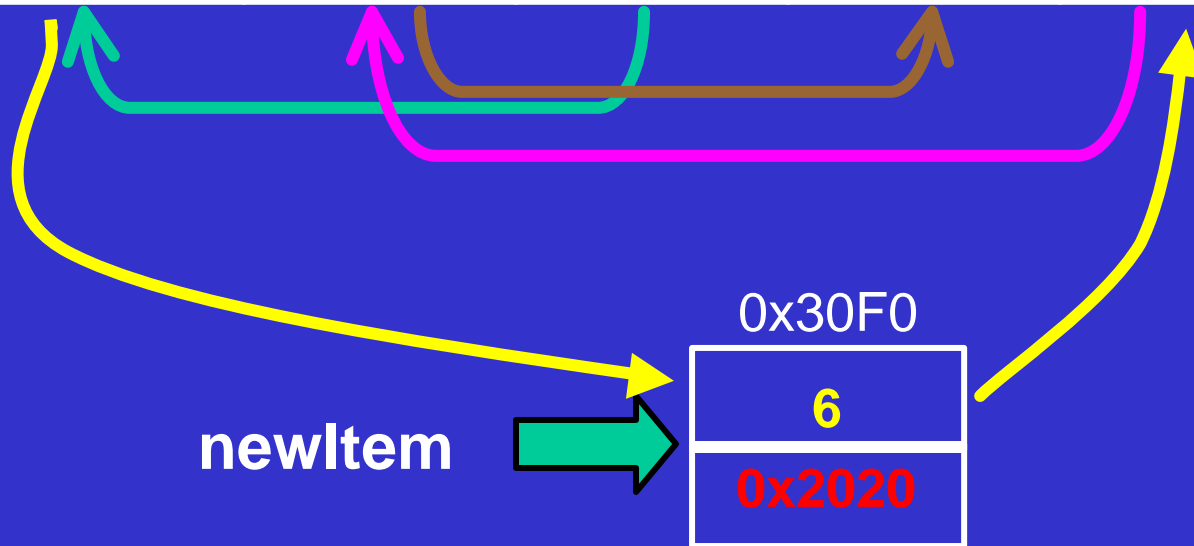


Method 2

start



0x2000	0x2008	0x2010	0x2018	0x2020
3	15	1	21	10
0x30F0	0x2018	0x2000	NULL	0x2008



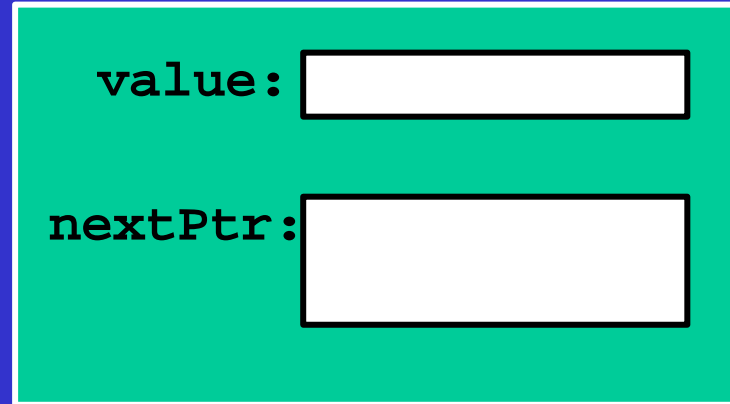
Advantages

- Only a little amount of memory needs to be allocated.
- Only a little amount of copying needs to be done.

```
struct NodeRec
{
    float value;
    struct NodeRec* nextPtr;
};

typedef struct NodeRec Node;
```

Node:



```
#ifndef NODE.H
#define NODE.H

struct NodeRec
{
    float value;
    struct NodeRec* nextPtr;
};

typedef struct NodeRec Node;

Node* makeNode(float item);

#endif
```

Make Node

- To make a **new node** for an item
 - take enough bytes from the **heap**
 - remember its **address** in memory
 - put the **item** in that location
 - set “**next**” link to NULL
 - **return** the node’s address

```
Node*
makeNode(float item)
{
    Node* newNodePtr = (Node*)malloc(sizeof(Node));

    if (newNodePtr == NULL) {
        fprintf(stderr, "Out of memory");
        exit(1);
    }
    else {
        newNodePtr->value = item;
        newNodePtr->nextPtr = NULL;
    }

    return newNodePtr;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

int main()
{
    float item;
    Node* firstNodePtr = NULL;
    Node* lastNodePtr = NULL;

    while (scanf("%f", &item) != EOF){
        if (firstNodePtr == NULL){
            firstNodePtr = makeNode(item);
            lastNodePtr = firstNodePtr;
        }
        else {
            lastNodePtr->nextPtr = makeNode(item);
            lastNodePtr = lastNodePtr->nextPtr;
        }
    }
}
```

firstNodePtr:

0x30F0

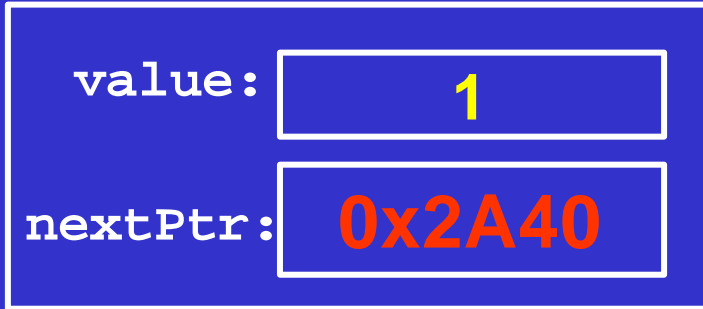
lastNodePtr:

0x2124

item:

6

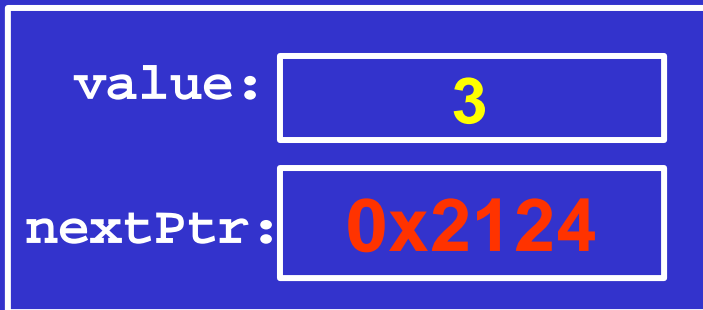
0x30F0



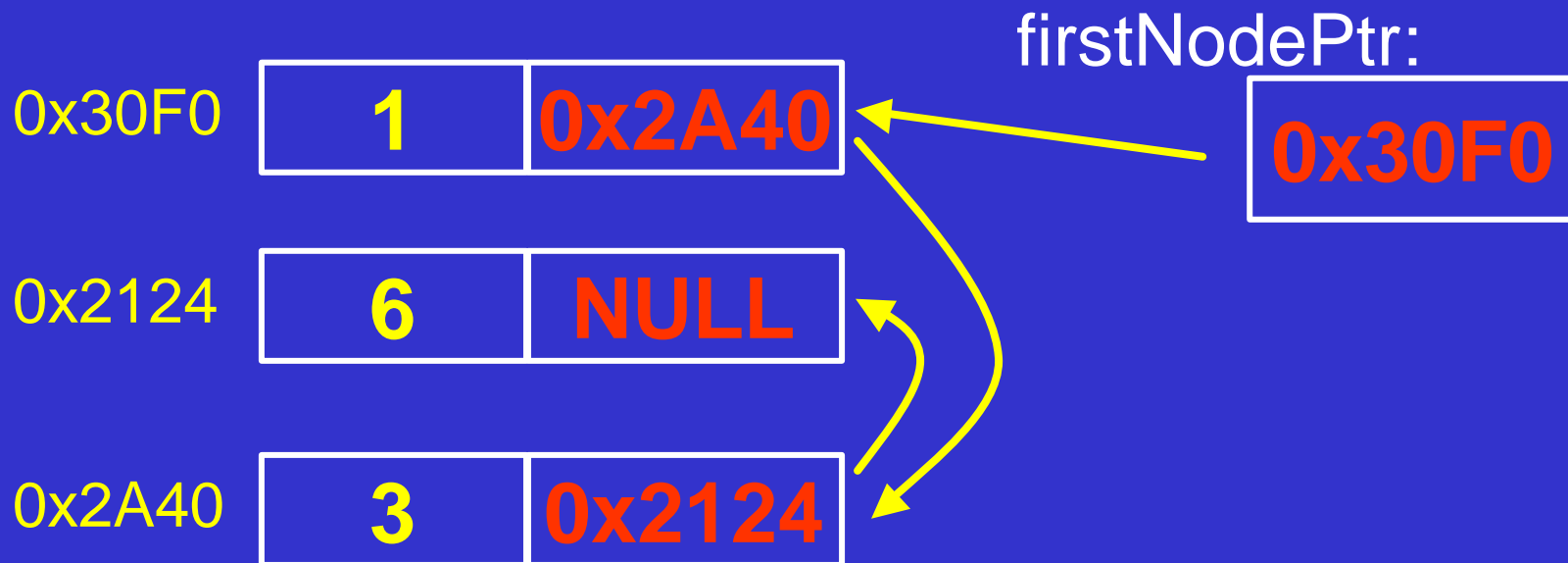
0x2124



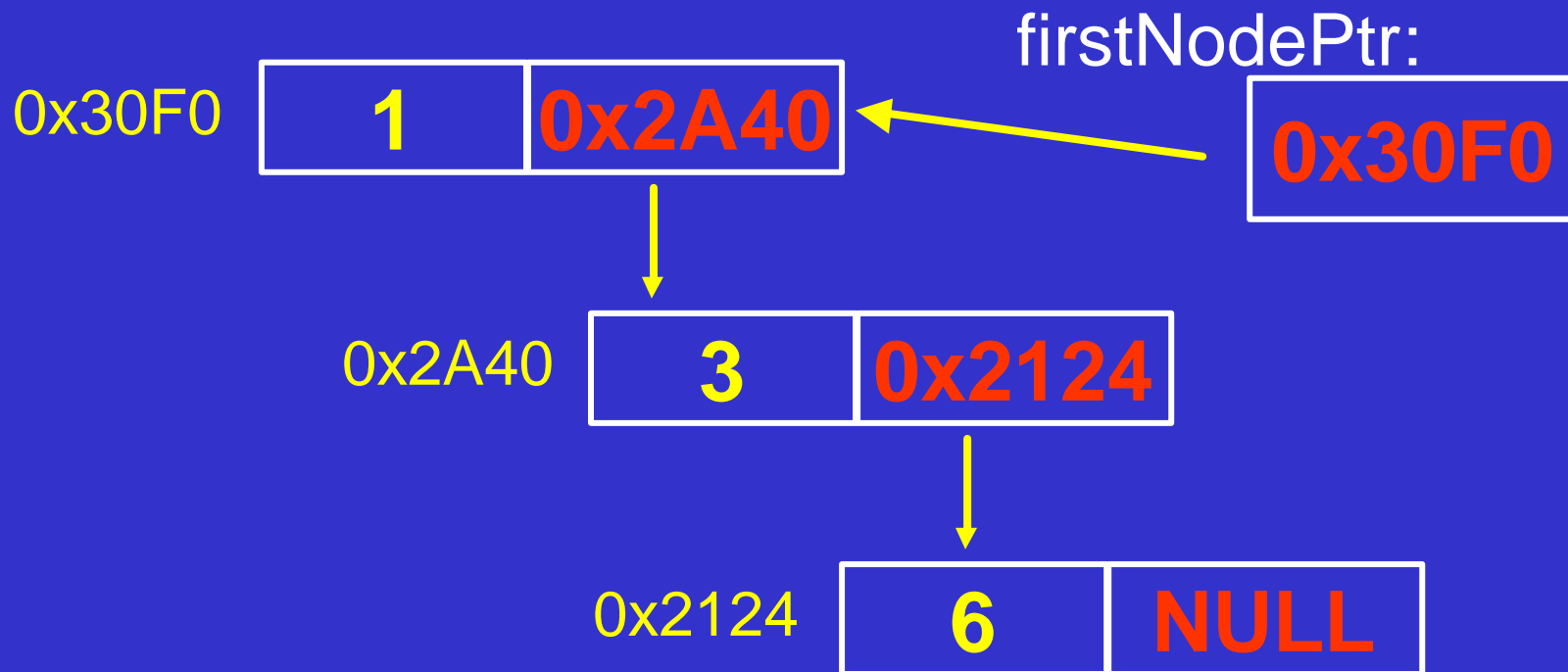
0x2A40



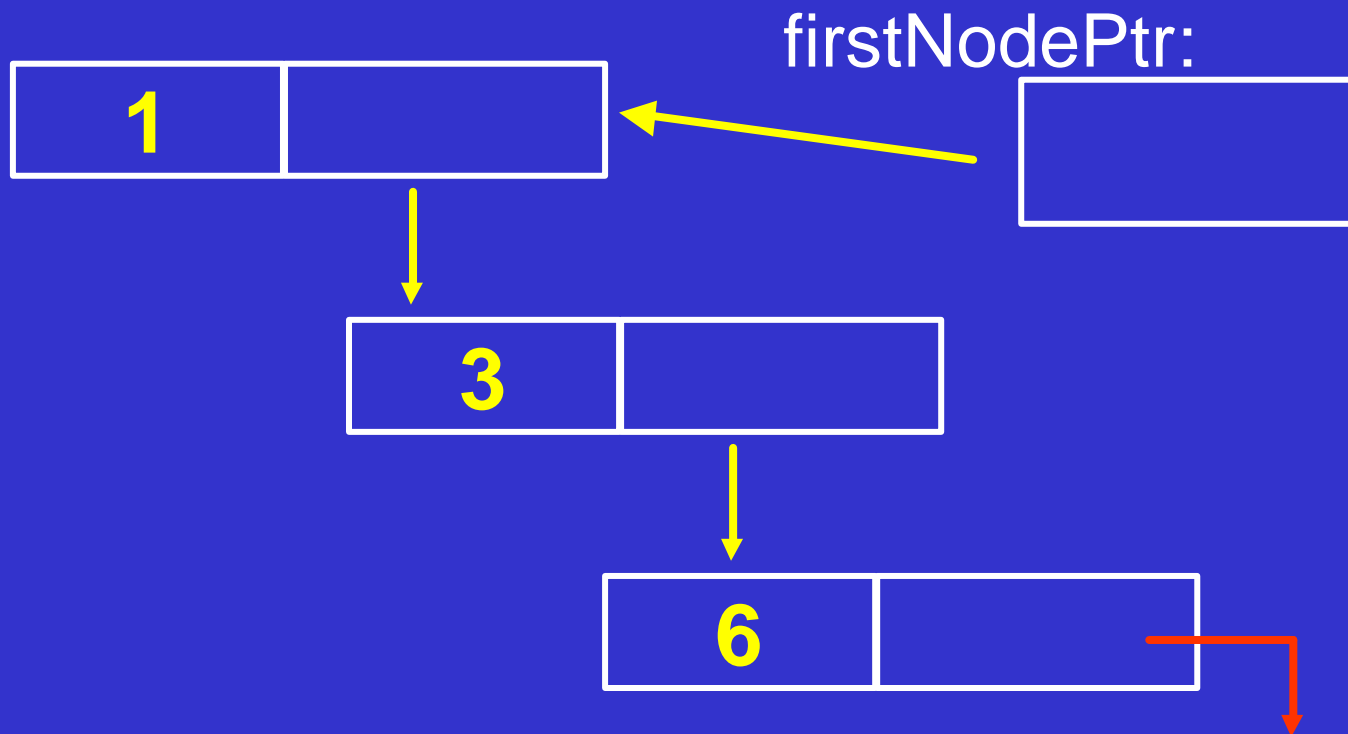
Linked Structure



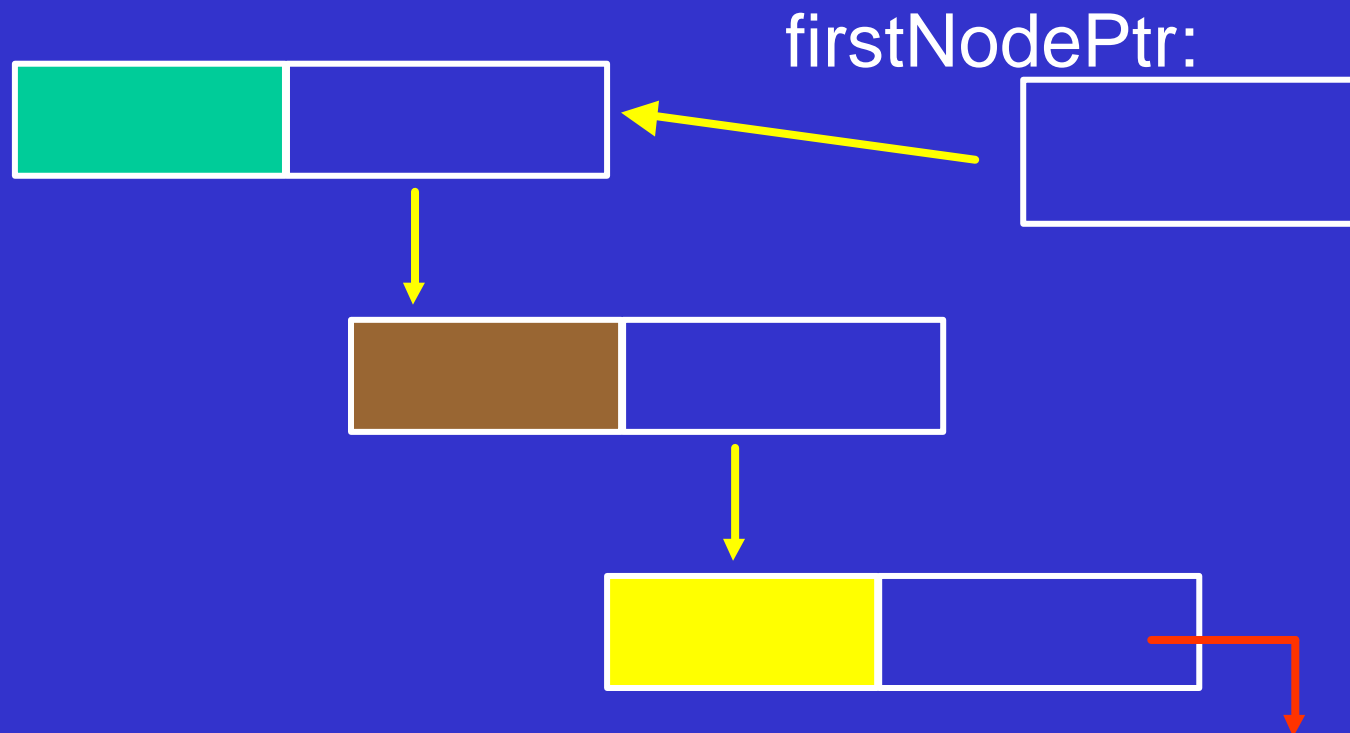
Linked Structure



Linked Structure



Linked Structure



Revision

- Node
- How to make a new Node
- Linked Structure

Preparation

- Read 3.1.6 in Kruse et al.