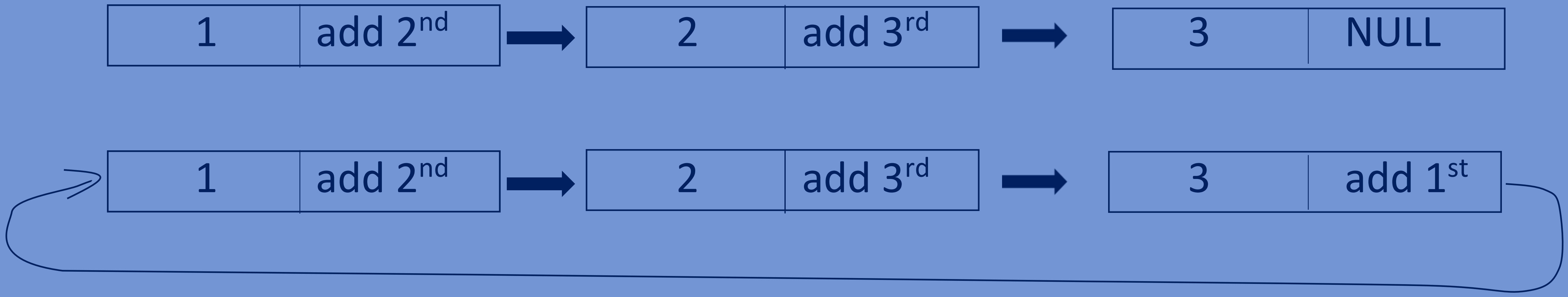
Data Structures

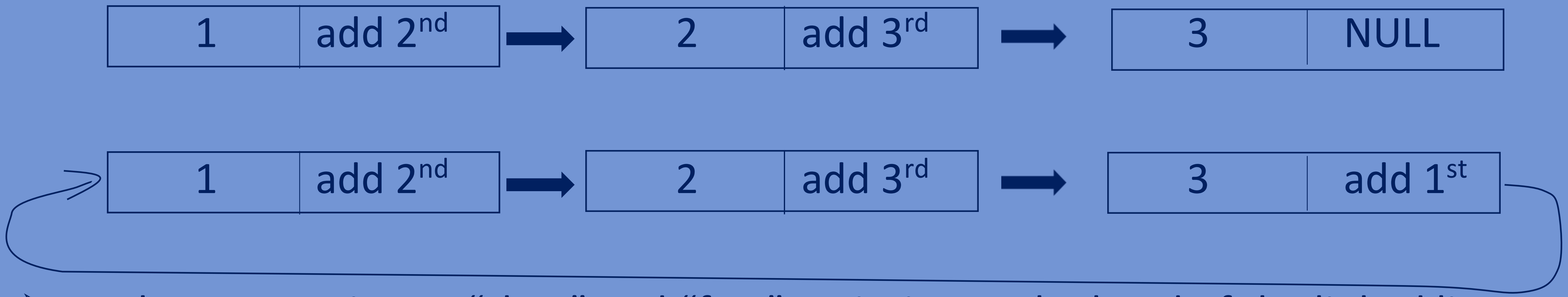
Cycle detection of a linked list

Cycle detection of a linked list:



- Check whether a linked list contains a cycle or not.
- Last node points to the other node of the same linked list.
- Every node of the linked list point to some other node of the same linked list.
- Use **Floyd's Cycle Finding Algorithm** for the detection of cycle in a linked list.

Floyd's Cycle-Finding Algorithm:

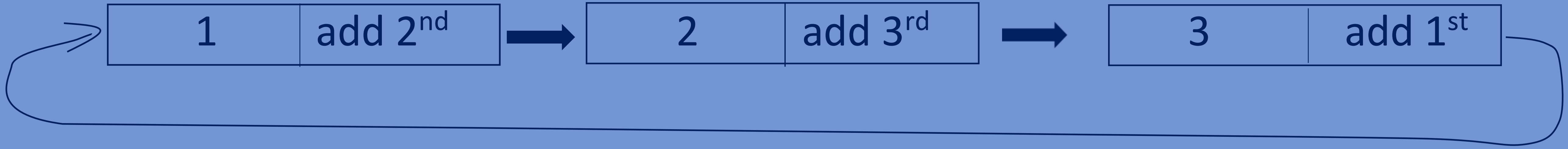


- Declare two pointers "slow" and "fast" ,pointing to the head of the linked list.
- After each iteration slow pointer moves forward by one node whereas fast pointer Moves forward two nodes at a time.

  slow=slow->next;

  fast=fast->next->next;
- the given linked list contains a loop or cycle , if at any point both the pointers refer to the same object(**slow==fast**).
- If the above condition is not met,it means the linked list doesn't have a loop.

Function for detecting of a cycle in linked list:



```
//cycle detection
int DetectCycle(lin_list *head){
    lin_list *slow=head;
    lin_list *fast=head;
    while(slow!=NULL && fast!=NULL && fast->next!=NULL ){
        fast=fast->next->next;
        slow=slow->next;
        if(slow==fast){
            return 1;
        }
    }
    return 0;
}
```

Whole program:

```
#include<stdio.h>
#include<stdlib.h>
//creating a node.
typedef struct lin_list{
    int data;
    struct lin_list *next;
}lin_list;
//inserting nodes
lin_list *insertnode(lin_list *head,int data) {
    lin_list *newnode=(lin_list*)malloc(sizeof(lin_list));
    newnode->data=data;
    newnode->next=head;
    head=newnode;
    return head;
}
//printing the linked list.
void PrintElements(lin_list *head){
    //base condition
    if(head==NULL){
        return;
    }
    printf("%d ",head->data);
    PrintElements(head->next);
}
//cycle detection
int DetectCycle(lin_list *head){
    lin_list *slow=head;
    lin_list *fast=head;
    while(slow!=NULL && fast!=NULL && fast->next!=NULL ){
        fast=fast->next->next;
        slow=slow->next;
        if(slow==fast){
            return 1;
        }
    }
    return 0;
}
```

```c
//main
int main(){
    lin_list *headA=NULL;
    //inserting elements into linked list
    headA=insertnode(headA,3);
    lin_list *temp=headA;
    headA=insertnode(headA,2);
    headA=insertnode(headA,2);
    headA=insertnode(headA,2);
    headA=insertnode(headA,1);
    PrintElements(headA);
    temp->next=headA;
    printf("\n%d\n",DetectCycle(temp));
      return 0;
}
```

<u>Output:</u>

```
1 2 2 2 3

1
```