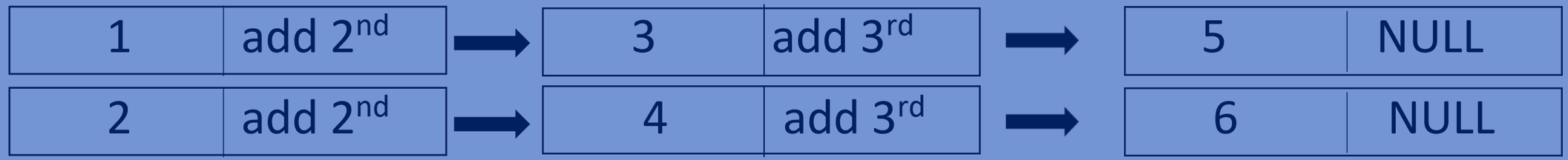
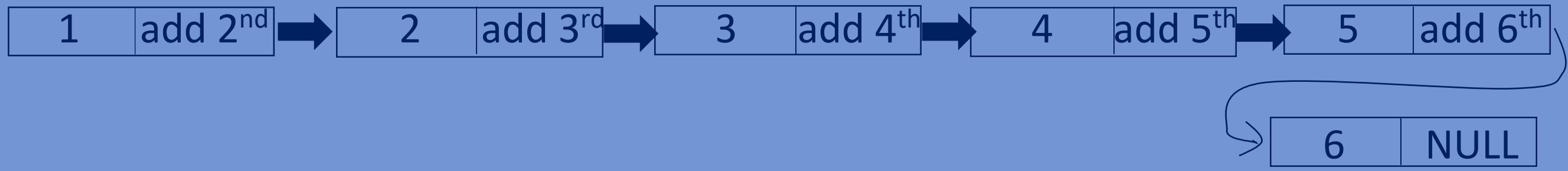
Data Structures

Merging of two sorted linked lists

Merging of two sorted linked lists:



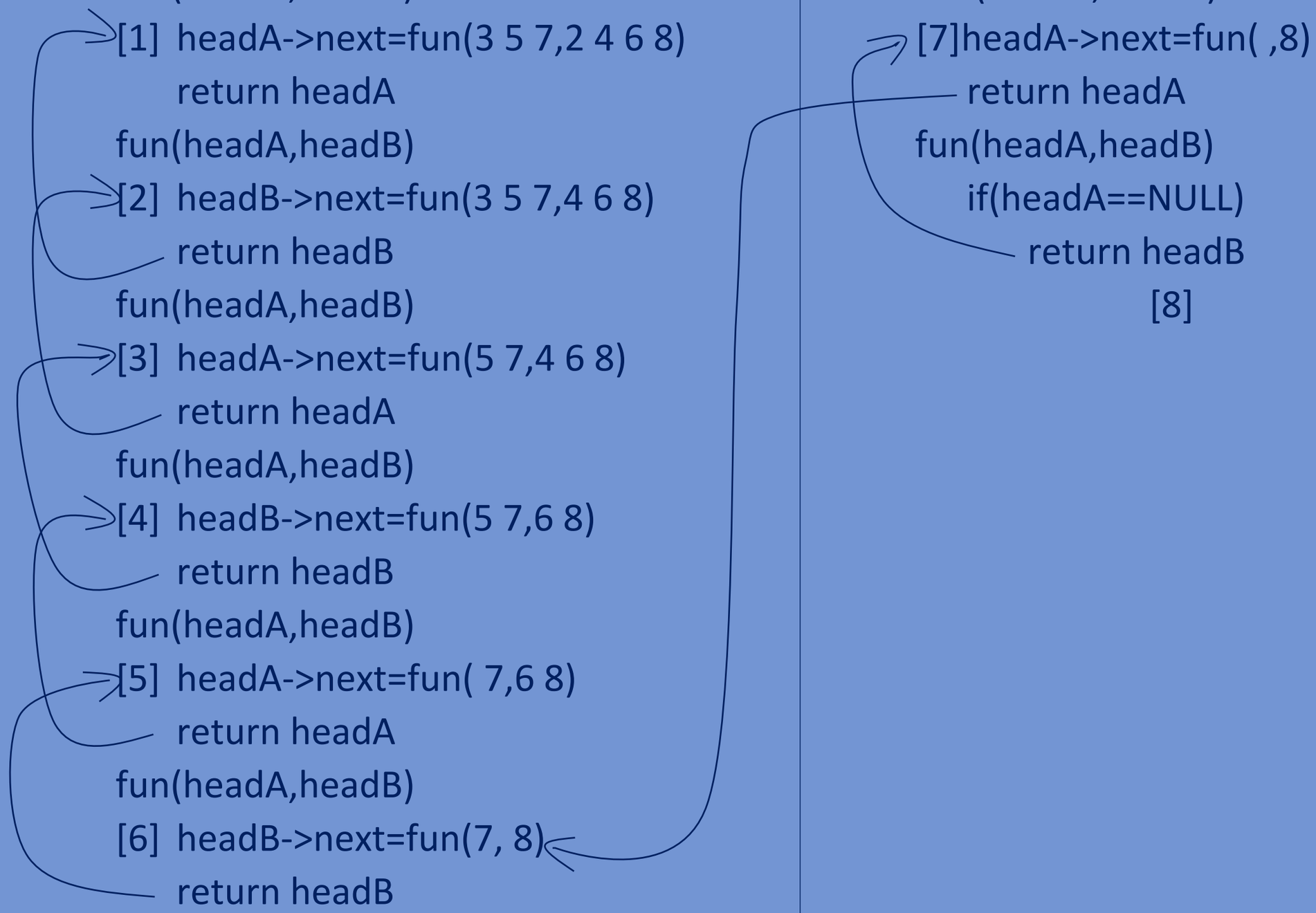


- Given two sorted linked lists, merge both the linked lists into a single linked list which should contain data in ascending order.
- Recursive approach.
- Base condition is if linked list A becomes NULL return linked list B and if Linked list B becomes NULL return linked list A.
- In other cases compare data of corresponding nodes of both the linked list.

- If the data in linked list A is greater than data in linked list B,then we call The function recursively with linkedlistB->next and linkedlist A and then Return linkedlist B else we call The function recursively with linkedlistA->next and linkedlist B and then Return linkedlist A.

Recursive function for merging of two sorted linked lists:

```
//merging of two sorted linnked lists
lin_list *mergeLinkedList(lin_list *headA,lin_list *headB){
    if(headA==NULL && headB!=NULL){
        return headB;
    }
    if(headA!=NULL && headB==NULL){
        return headA;
    }
    if(headA->data > headB->data){
        headB->next=mergeLinkedList(headA,headB->next);
        return headB;
    }
    else{
        headA->next=mergeLinkedList(headA->next,headB);
        return headA;
    }
}
```


headA=1 3 5 7 , headB=2 4 6 8
fun(headA,headB)
[1] headA->next=fun(3 5 7,2 4 6 8)
return headA
fun(headA,headB)
[2] headB->next=fun(3 5 7,4 6 8)
return headB
fun(headA,headB)
[3] headA->next=fun(5 7,4 6 8)
return headA
fun(headA,headB)
[4] headB->next=fun(5 7,6 8)
return headB
fun(headA,headB)
[5] headA->next=fun(7,6 8)
return headA
fun(headA,headB)
[6] headB->next=fun(7, 8)
return headB
fun(headA,headB)
[7]headA->next=fun(,8)
return headA
fun(headA,headB)
if(headA==NULL)
return headB
[8]

Whole Program:

```c
#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);
}
//merging of two sorted linnked lists
lin_list *mergeLinkedList(lin_list *headA,lin_list *headB){
    if(headA==NULL && headB!=NULL){
        return headB;
    }
    if(headA!=NULL && headB==NULL){
        return headA;
    }
    if(headA->data > headB->data){
        headB->next=mergeLinkedList(headA,headB->next);
        return headB;
    }
    else{
        headA->next=mergeLinkedList(headA->next,headB);
        return headA;
    }
}
//main
int main(){
    lin_list *headA=NULL;
    lin_list *headB=NULL;
```

```
    //inserting elements into linked list A and B
    headA=insertnode(headA,7);
    headA=insertnode(headA,5);
    headA=insertnode(headA,3);
    headA=insertnode(headA,1);

    headB=insertnode(headB,8);
    headB=insertnode(headB,6);
    headB=insertnode(headB,4);
    headB=insertnode(headB,2);
    PrintElements(headA);printf("\n");
    PrintElements(headB);printf("\n");
    headA=mergeLinkedList(headA,headB);
    PrintElements(headA);
    return 0;
}
```

Output:

```
1 3 5 7
2 4 6 8
1 2 3 4 5 6 7 8
```