Computer Programming 1 Lab

2020-12-10

□□

Outline

- Quick Sort
- Linked List
- Exercise 10
- Homework

Quick Sort

Quick Sort

Your Sort ?

```
#define N 10

int array[N];
for(int i=0;i<N;i++)
    for(int j=0;j<N;j++)
        if(array[i] > array[j])
            swap(array[i], array[j]);
```

Quick Sort

How to use

- you need to #include <stdlib>

```c
#define N 10

int main(void){
    int array[N];
    // Only one line !!!
    qsort((void *)array, N, sizeof(int), compare);
}
```

Quick Sort

Compare Function

- you need to #include <stdlib>

```
int compare(const void *a, const void *b){
    int A = *(int *)a;
    int B = *(int *)b;
    if(A < B)
        return -1; //□□ -1 □□ a < b
    else if(A == B)
        return 0; //□□ 0 □□ a = b
    else
        return 1; //□□ 1 □□ a > b
}
```

Quick Sort

Compare Function

- you need to #include <stdlib>

```c
#define N 10

int compare(const void *a, const void *b){
    int A = *(int *)a;
    int B = *(int *)b;
    // Oh !!!
    return A - B;
}

int main(void){
    int array[N];
    qsort((void *)array, N, sizeof(int), compare);
}
```

Quick Sort

Compare Function

- you need to #include <stdlib>

```c
#define N 10

int compare(const void *a, const void *b){
    // Oh my Jesus
    return *(int *)a - *(int *)b;
}

int main(void){
    int array[N];
    qsort((void *)array, N, sizeof(int), compare);
}
```

Quick Sort

Quick Sort in String

```c
#define N 10

int cmp(const void *a, const void *b){
    return strcmp((char *)a, (char *)b);
}

int main(void){
    char Dic[N][20];
    qsort((void *)Dic, N, sizeof(Dic[0]), cmp);
}
```

Quick Sort

Quick Sort in Struct

```c
#define N 10
typedef struct pikachu{
    int Attack;
    int Defense;
} Pikachu;

int cmp(const void *a, const void *b){
    Pikachu A = *(Pikachu *)a;
    Pikachu B = *(Pikachu *)b;
    return A.attack - B.attack;
}

int main(void){
    Pikachu Pikachus[N];
    qsort((void *)Pikachus, N, sizeof(Pikachus[0]), cmp);
}
```

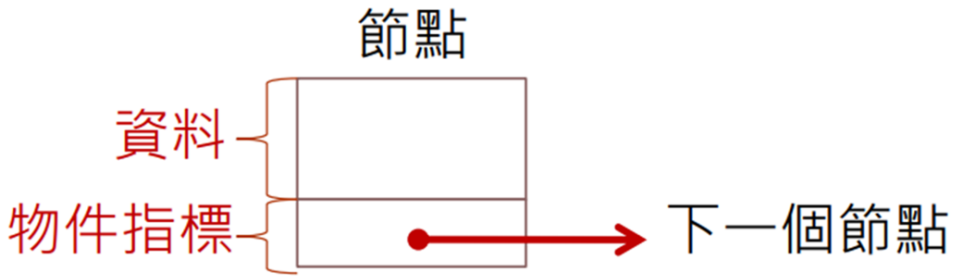
Linked List

鏈結串列

Linked List

Node (節點)

- 構成串列的最小單位



節點 = 資料+ 物件指標

Linked List

Node □□

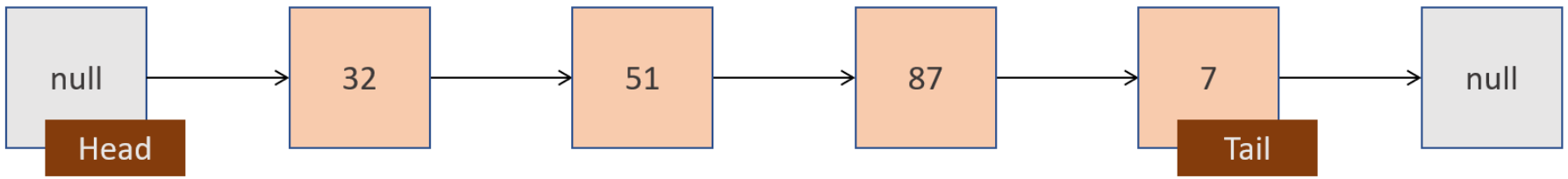
- □□□□□□□□□

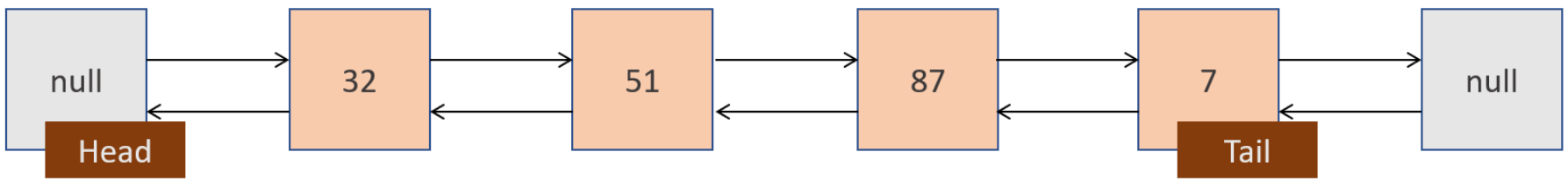
```
typedef struct node Node;

struct node{
    int value;
    Node *next;
};
```

Linked List

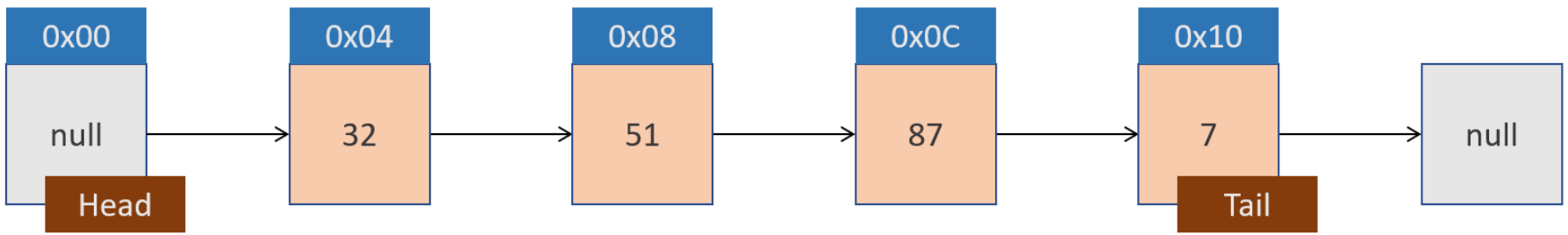
```c
typedef struct node Node;

struct node{
    int value;
    Node *next;
};

Node head;

int main(void){
    // create a node
    Node *tmp = malloc(sizeof(Node));
    tmp -> value = 10;
    tmp -> next = NULL;
    // add first node
    head->next = tmp;
    printf("%d\n", head.next->value);
    // clear
    free(head.next);
}
```

Linked List

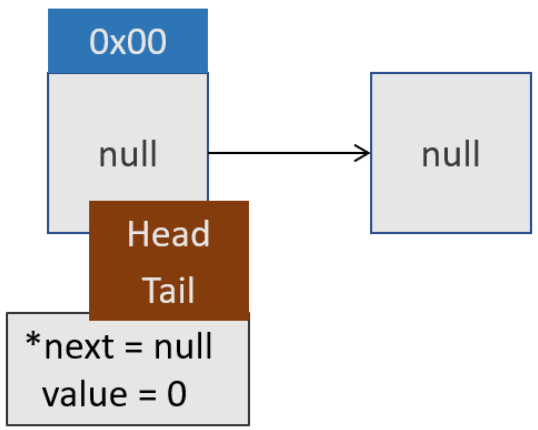




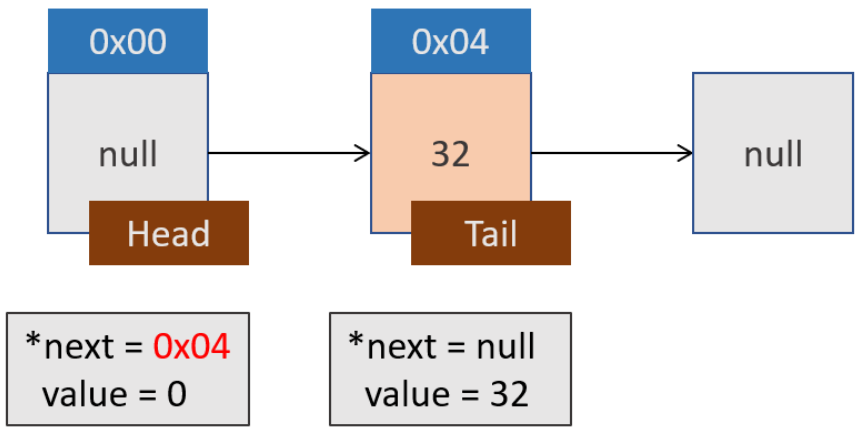
Linked List



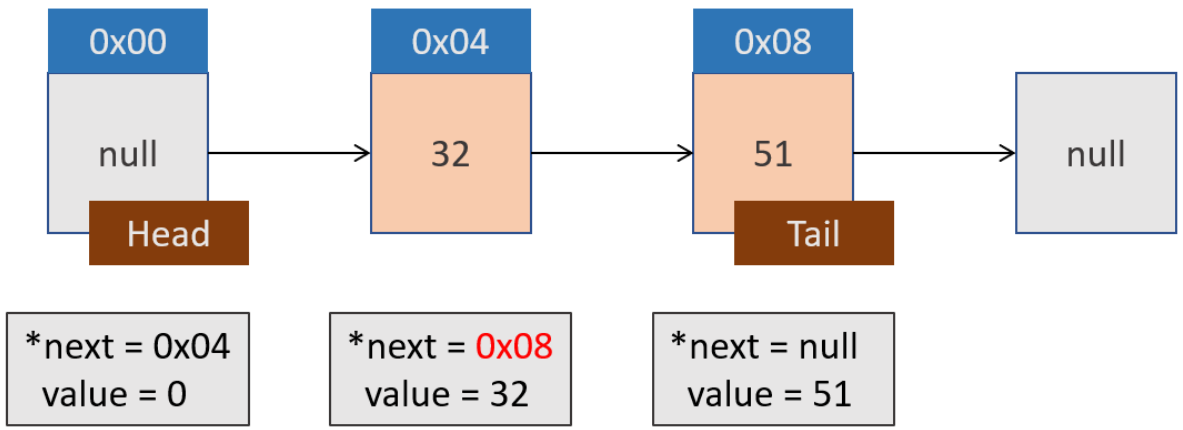
Linked List



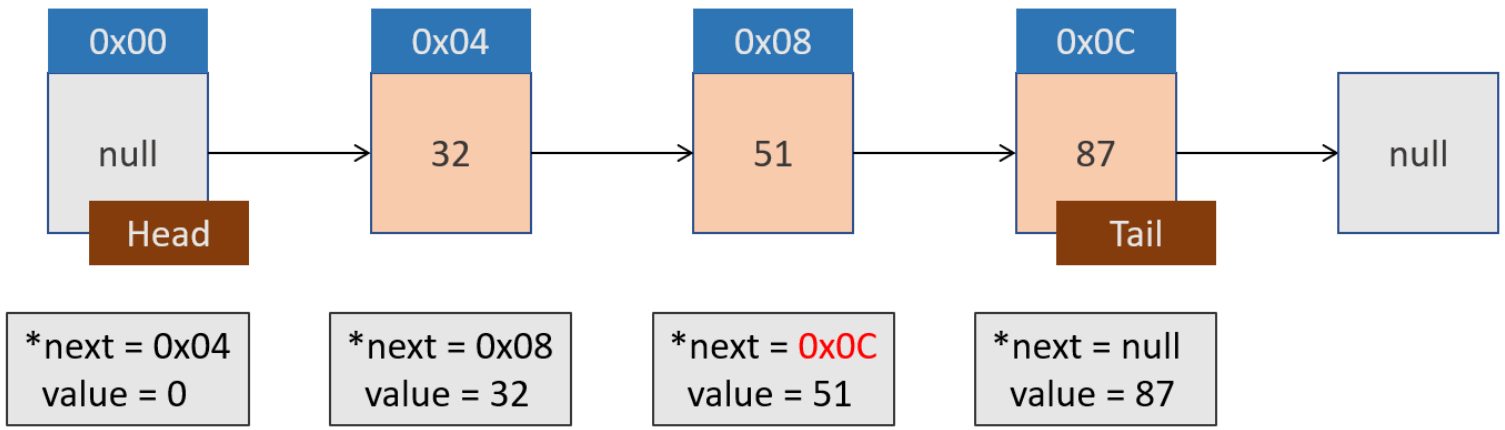
Linked List


0x00
null
Head
0x04
32
Tail
null
*next = 0x04
value = 0
*next = null
value = 32

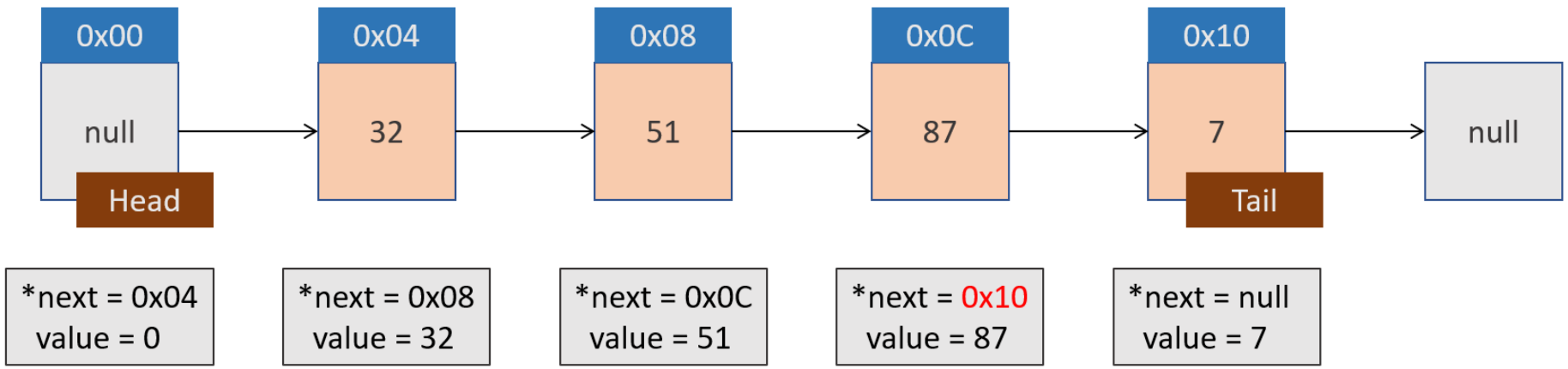
Linked List


0x00
null
Head
0x04
32
0x08
51
Tail
null
*next = 0x04
value = 0
*next = 0x08
value = 32
*next = null
value = 51

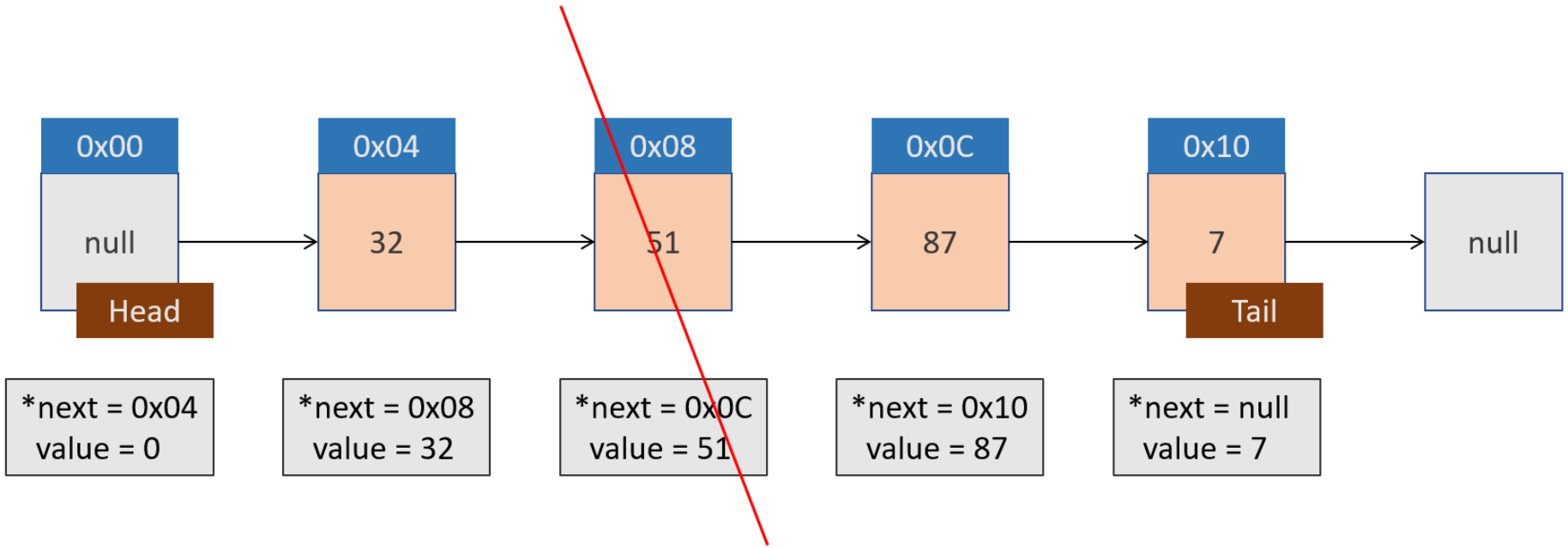
Linked List


0x00
null
Head
0x04
32
0x08
51
0x0C
87
Tail
null
*next = 0x04
value = 0
*next = 0x08
value = 32
*next = 0x0C
value = 51
*next = null
value = 87

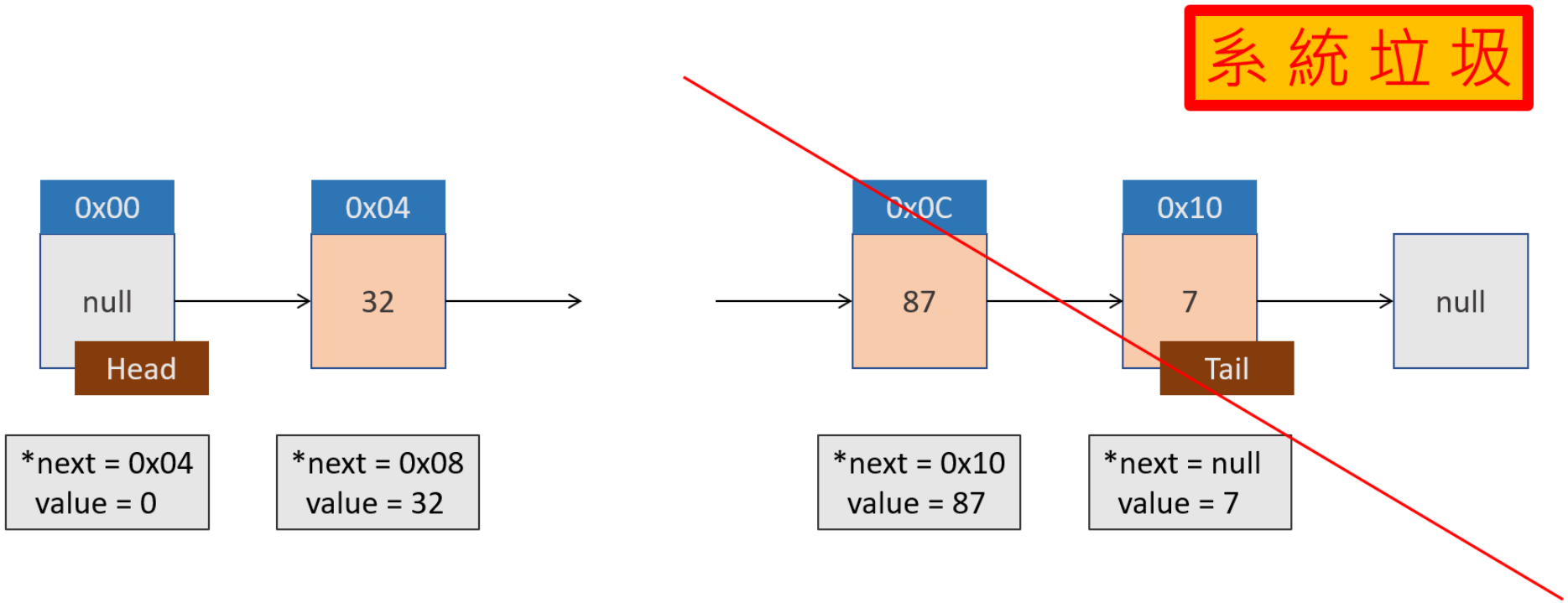
Linked List


0x00
null
Head
0x04
32
0x08
51
0x0C
87
0x10
7
Tail
null
*next = 0x04
value = 0
*next = 0x08
value = 32
*next = 0x0C
value = 51
*next = 0x10
value = 87
*next = null
value = 7

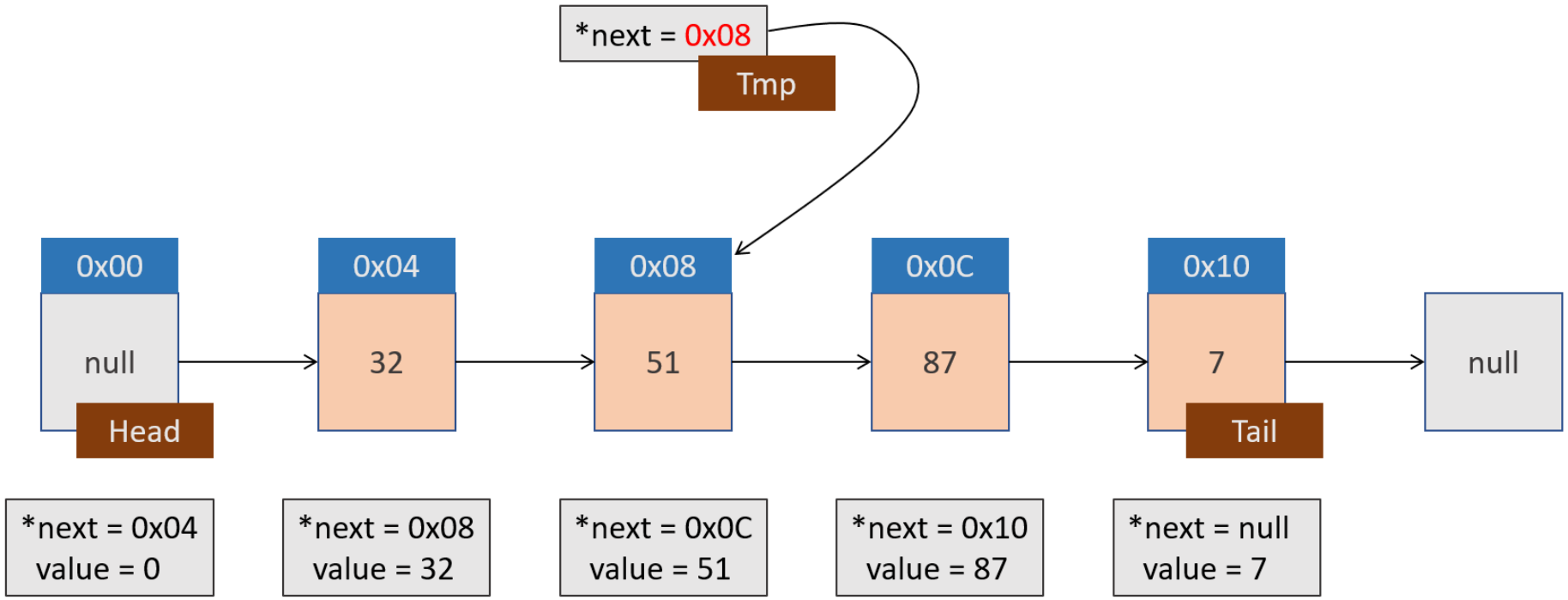
Linked List - Delete



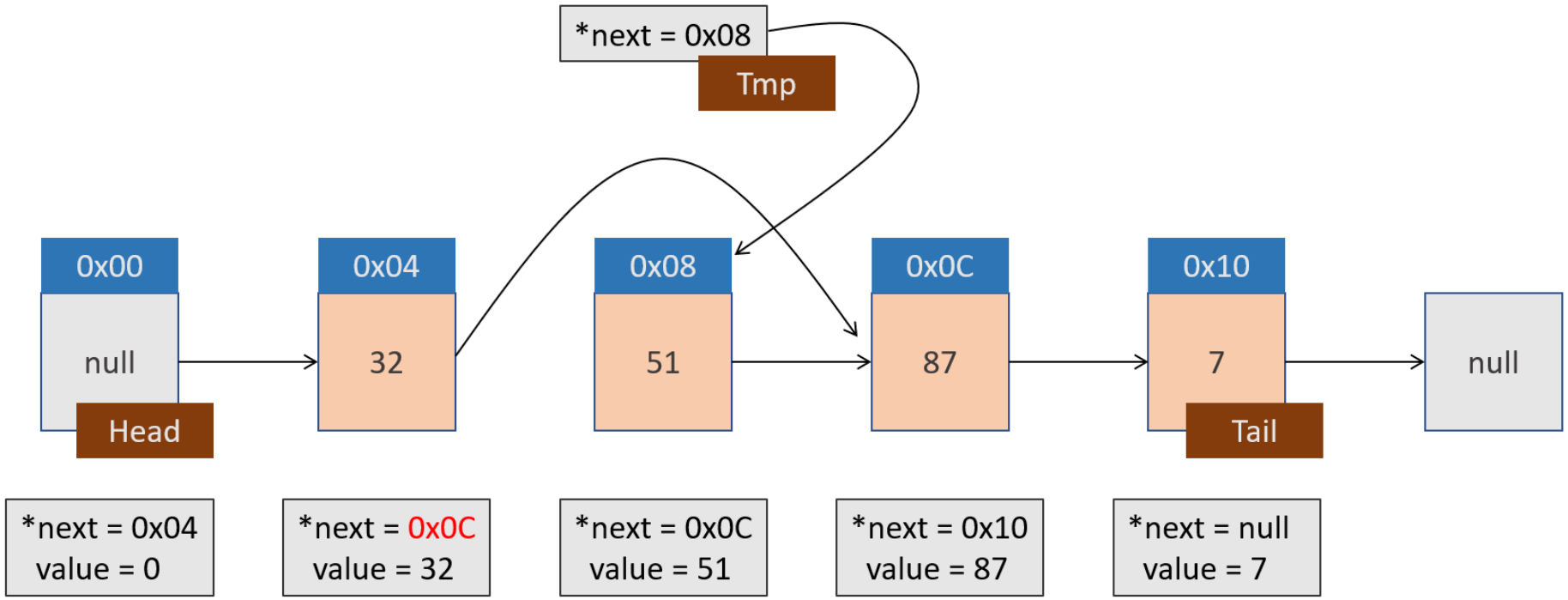
Linked List - Delete



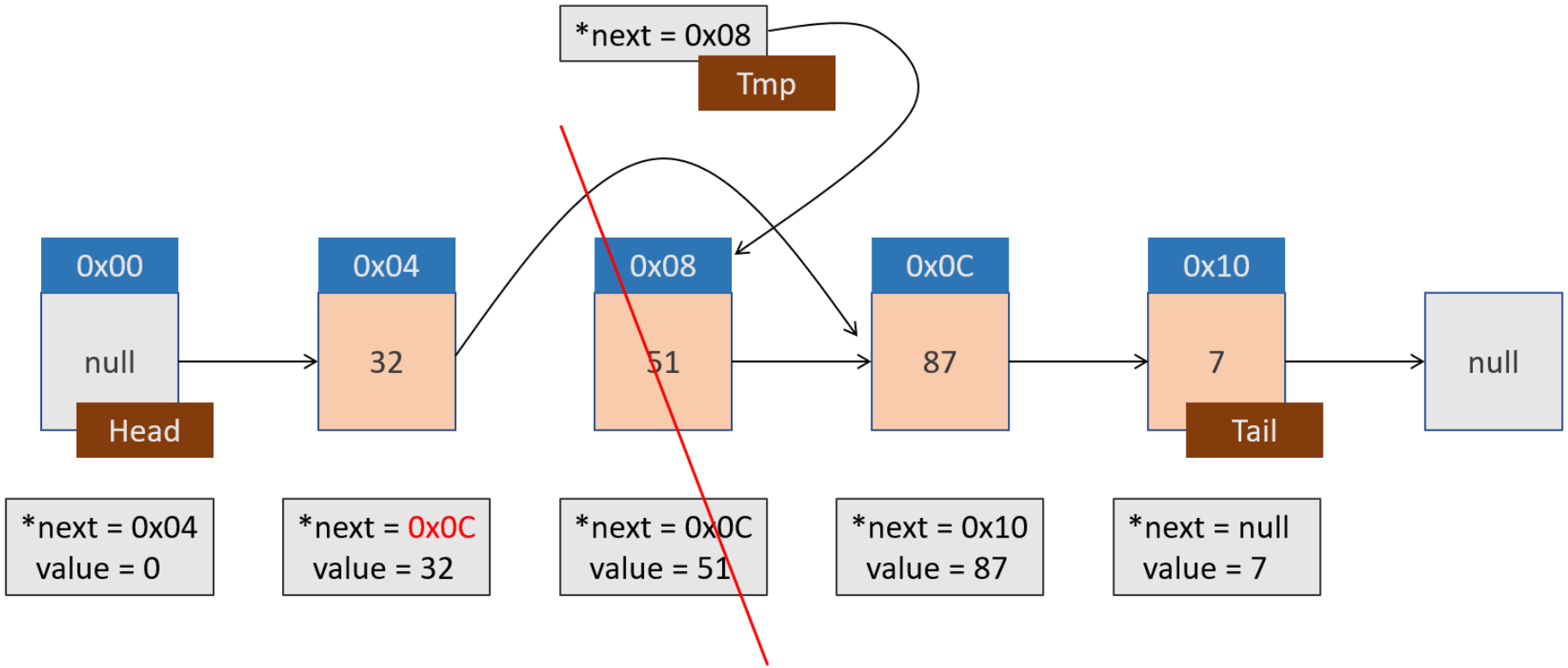
Linked List - Delete



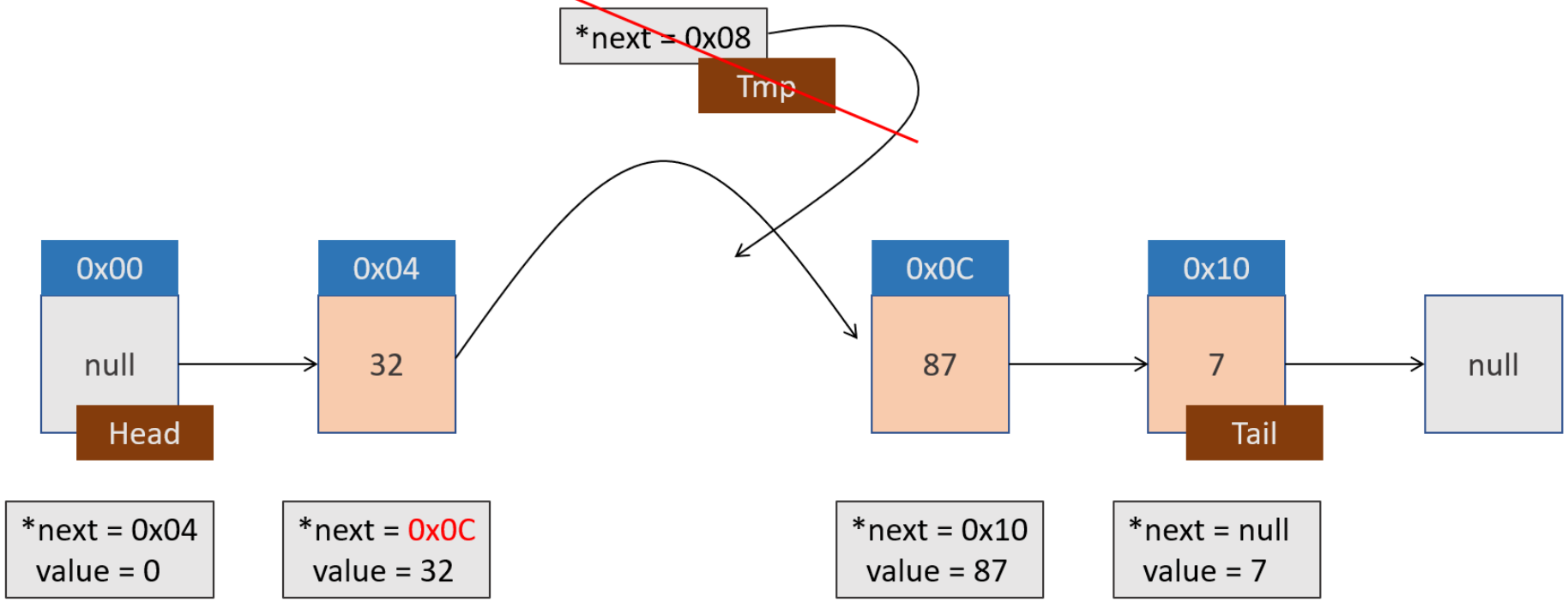
Linked List - Delete



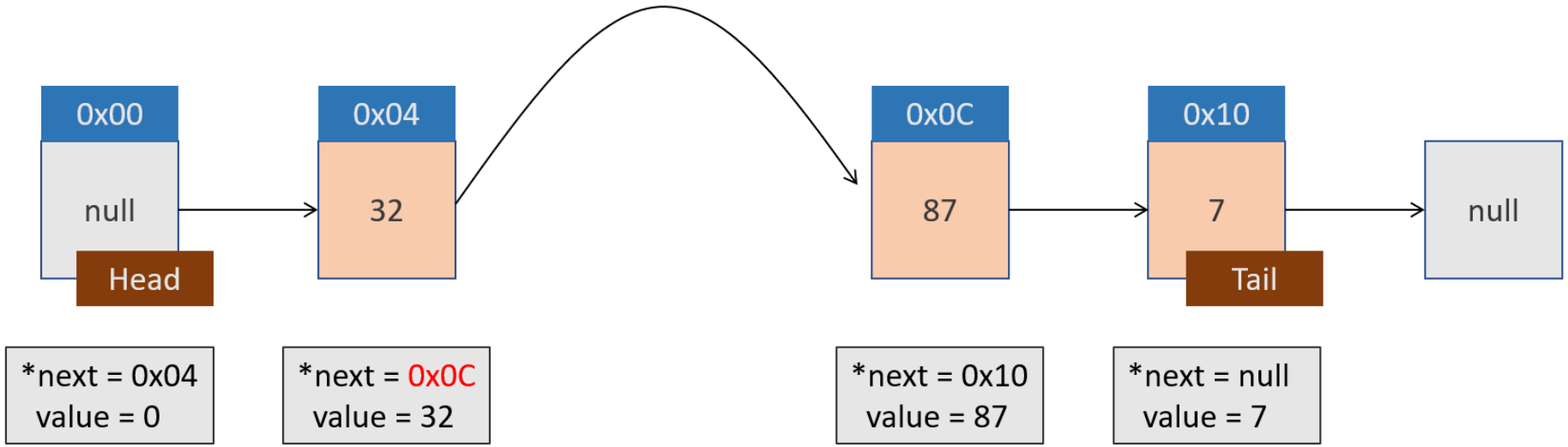
Linked List - Delete



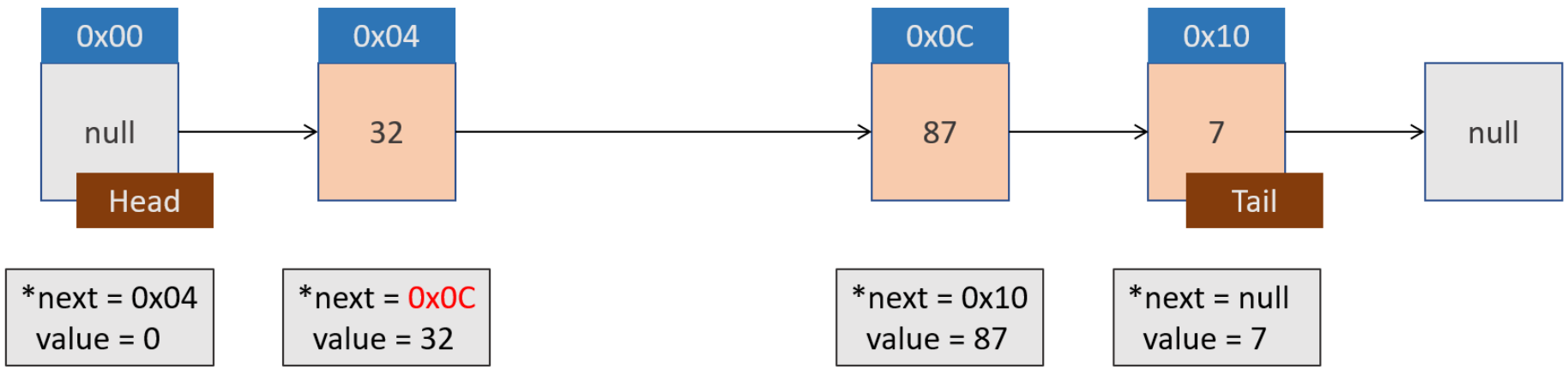
Linked List - Delete



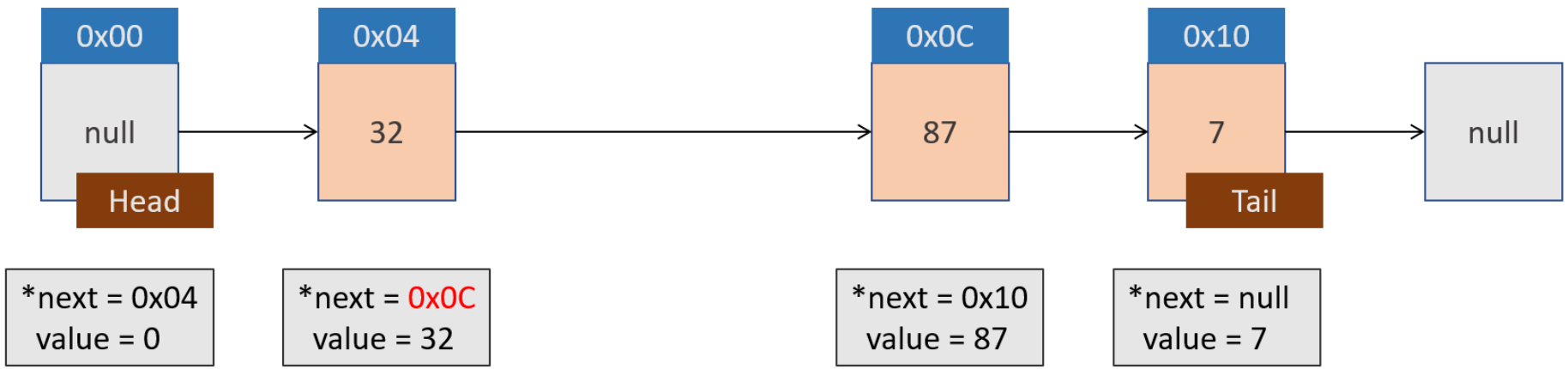
Linked List - Delete



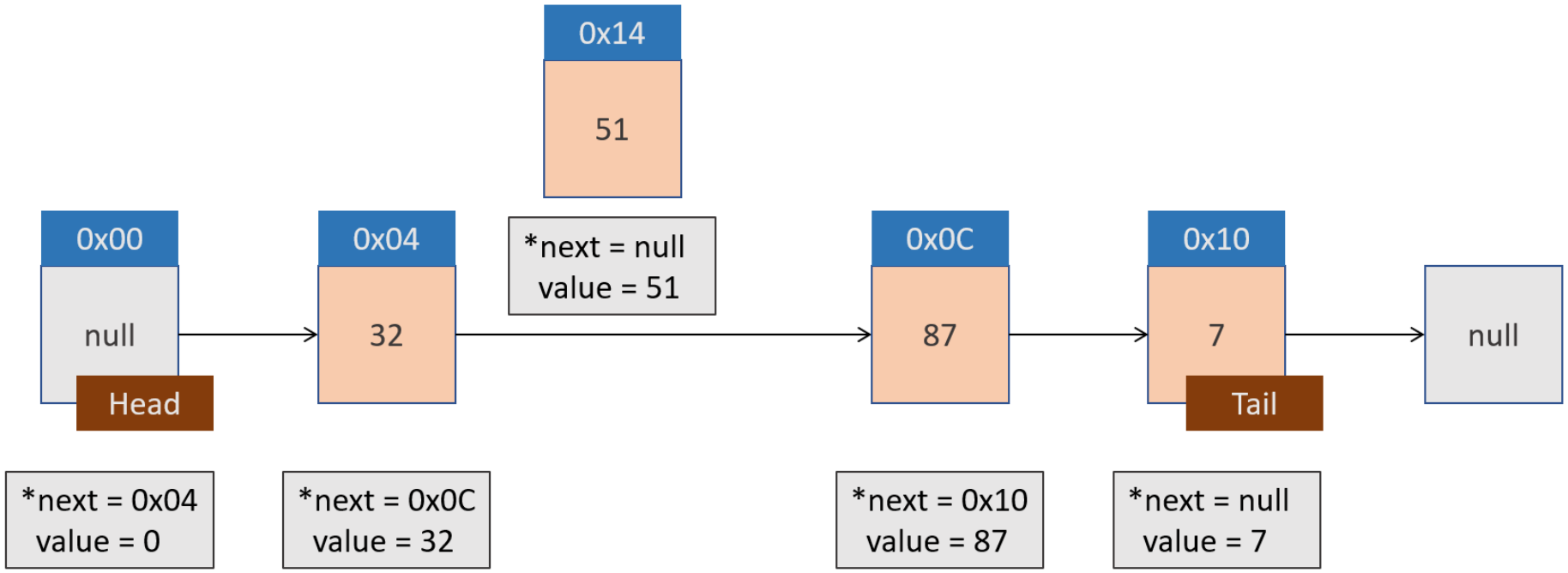
Linked List - Delete



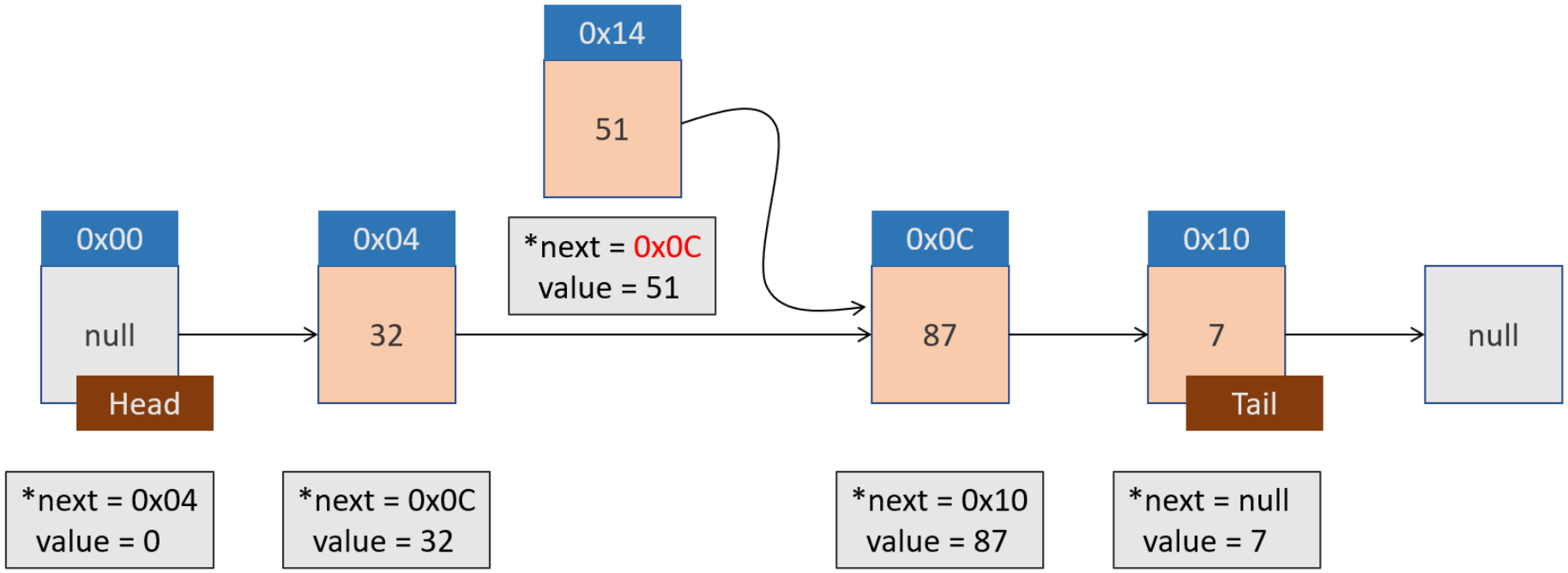
Linked List - Insert



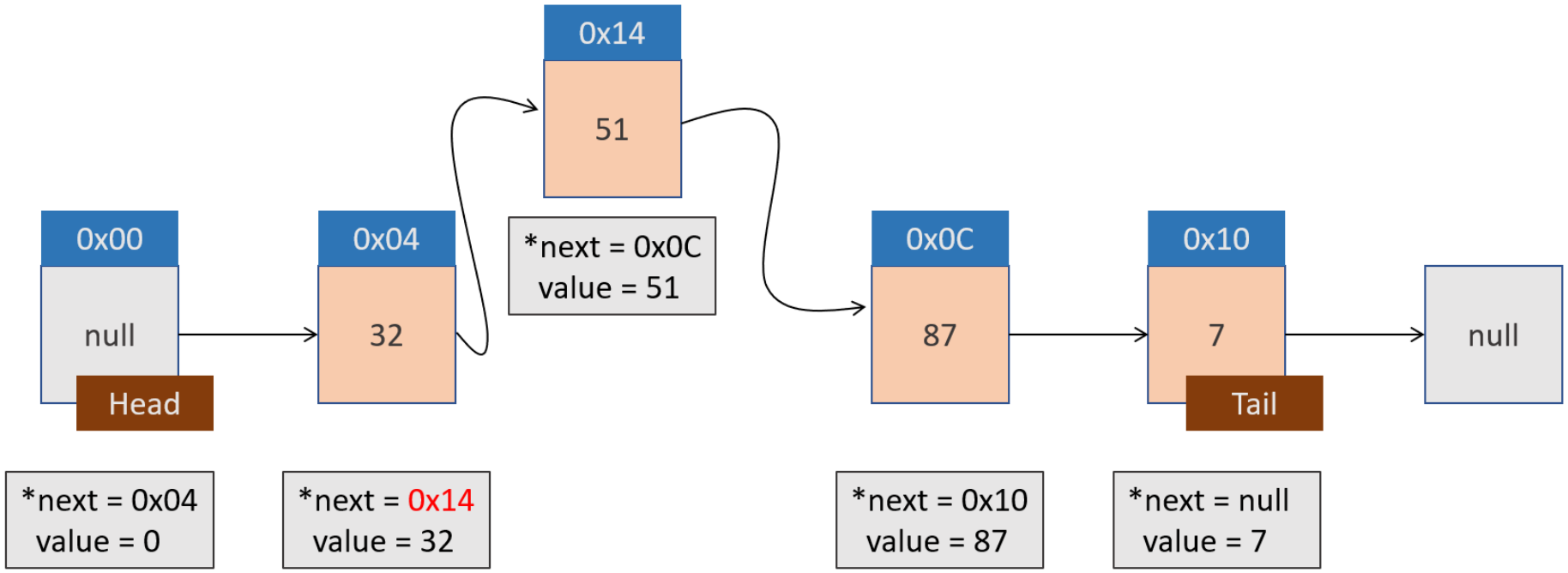
Linked List - Insert



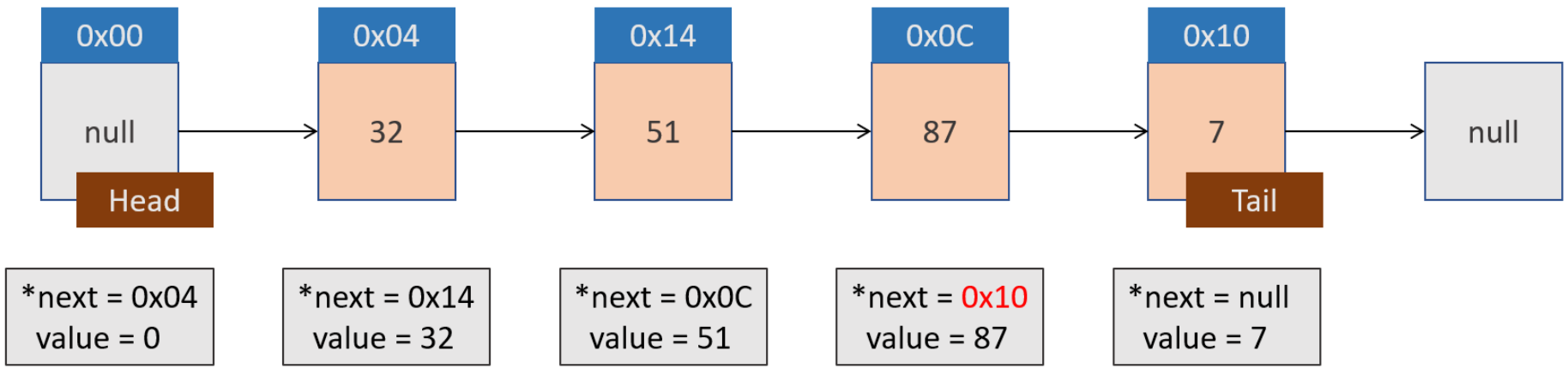
Linked List - Insert



Linked List - Insert

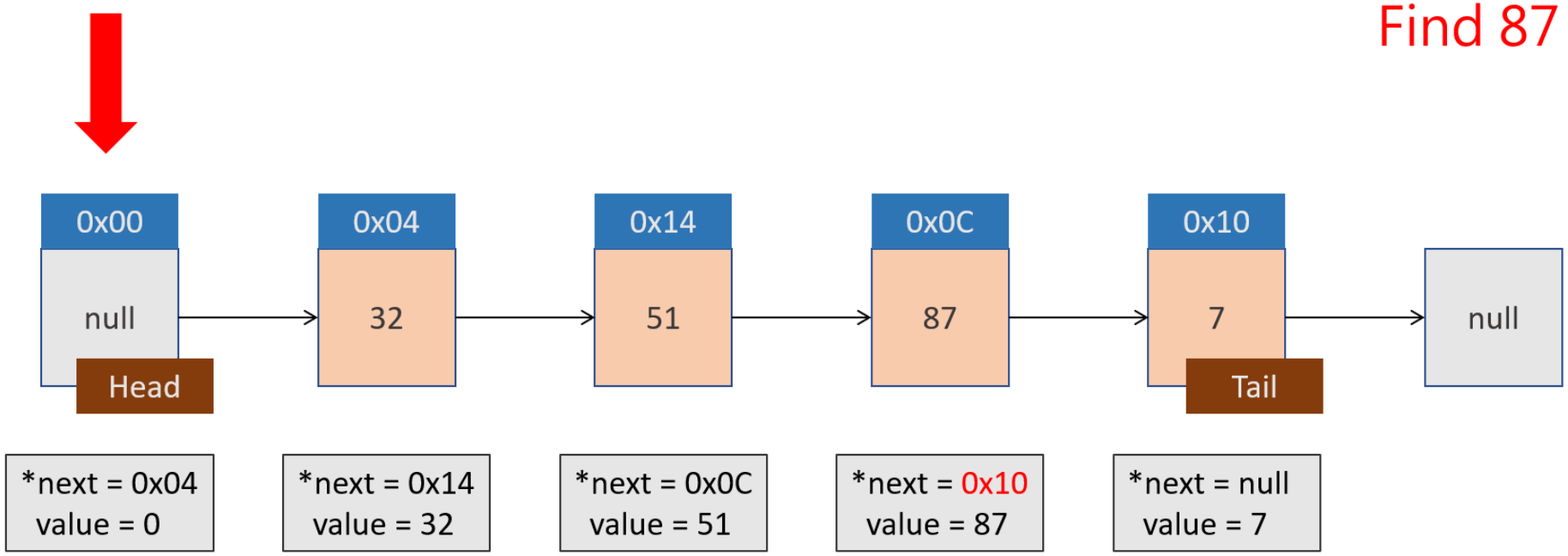


Linked List - Insert



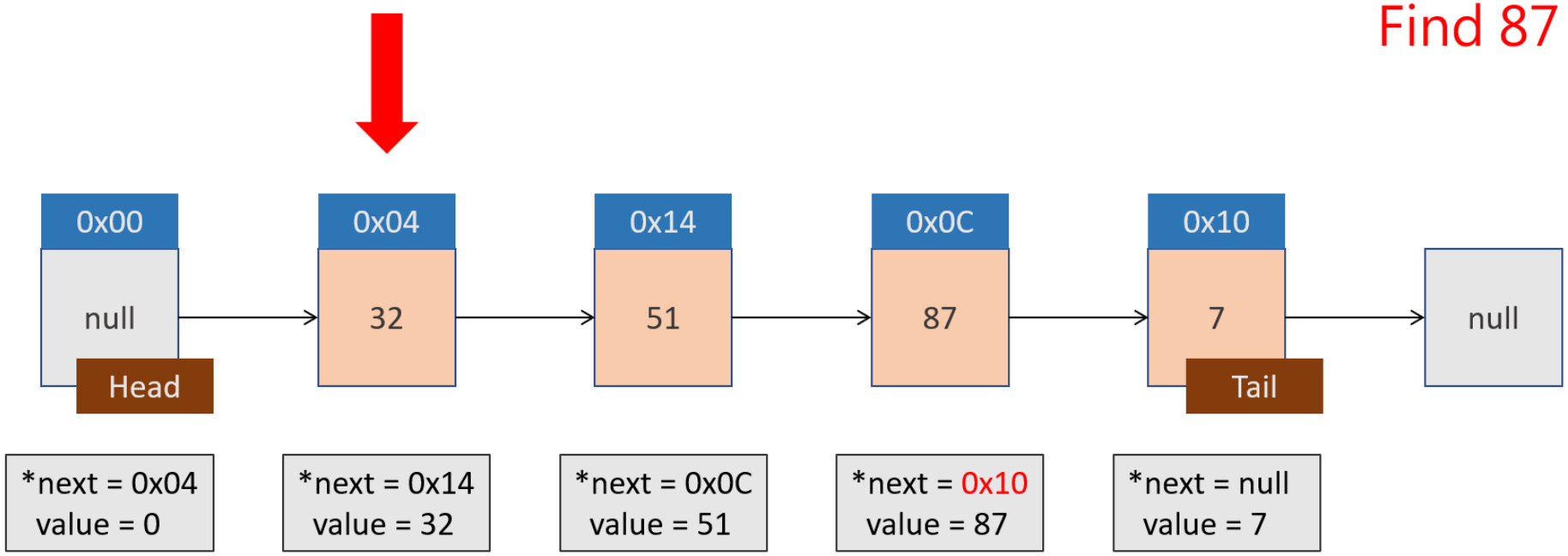
Linked List – Find Element

Find 87



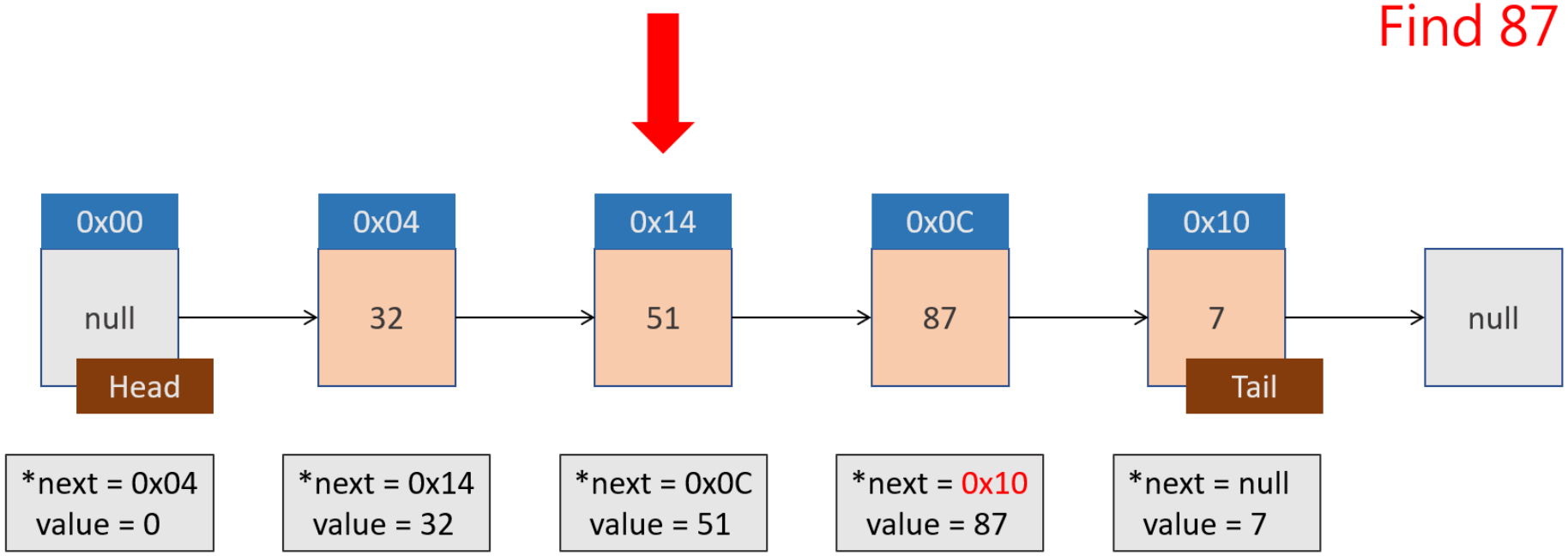
Linked List – Find Element

Find 87

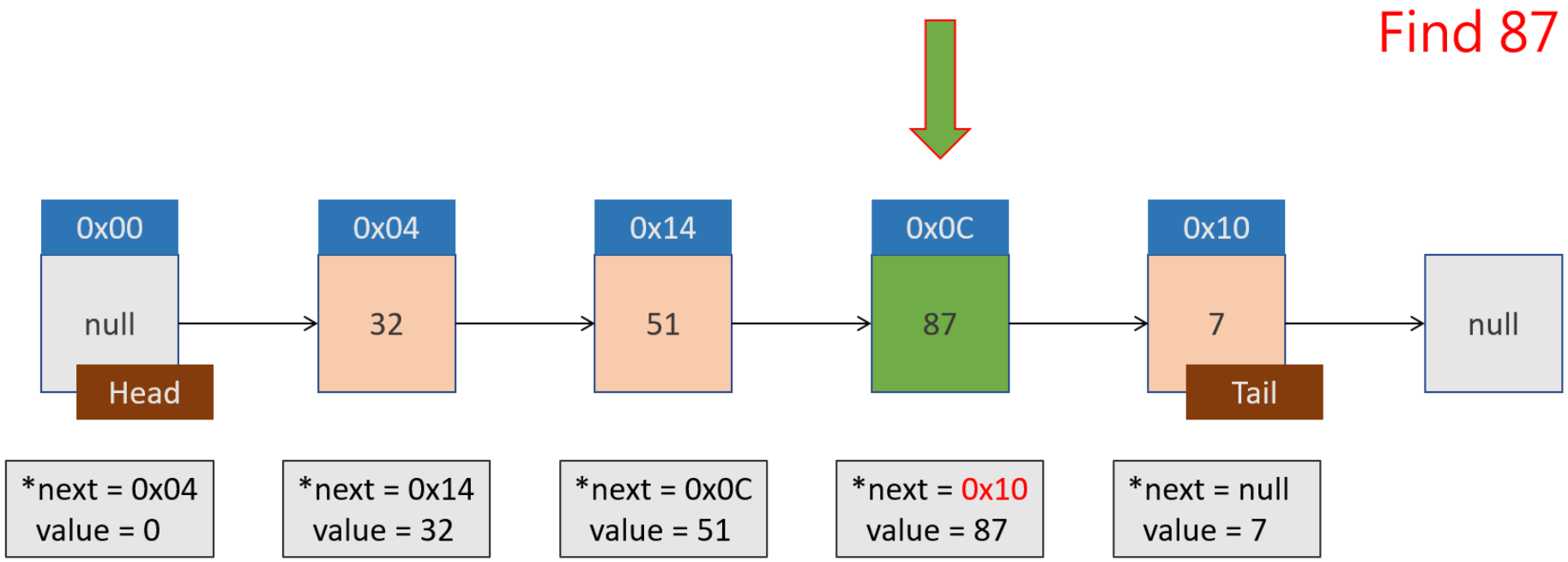


Linked List – Find Element

Find 87



Linked List – Find Element



陣列 VS 串列

陣列	串列
占用連續的記憶體空間	可以非連續
各元素型態皆相同	各節點型態不必一定相同
插入、刪除元素麻煩	方便
無法動態增加、刪除空間	可以
可支援循序及隨機存取	僅支援循序存取
可靠度高	低，因為指標斷裂，資料就 遺失
循序存取速度快	慢，因為必須先讀取指標

Exercise 10

Pokemon GO

- Input:
  □□□□□□□□□□□□□□□□□□□□
  □□□□□□□□□□□□□□□□□□□□
- Output:
  □□□□□□□□□□□□□□□□□□□□□□ OJ

Homework

- [illegible]

Any Question?