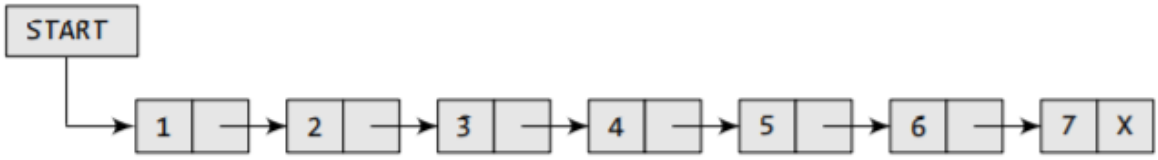
DSA Exp no. 1: SINGLY LINKED Lists




```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct node
{
        int data;
        struct node *next;
};
```

Function declare and define

Function1

{

}

Function2

{

}

VOID MAIN

{

Struct node * Start=NULL;

Int ch;

//Function call

do

{

printf("Enetr 1 – display 2- insert in beg 3- insert at end 4- insert after a node 5- delete first node 6- dlete last node 7- delete given node 8-exit");

scanf("%d",&ch);

```
switch(ch)
{
Case 1: display(Start);
        Break;
Case 2: Start=Insert_beg(Start);
        display(Start);
        break;
Case3:
}                                   //switch case
}while(ch!=8);                      //do-while loop
getch();
}
```

Traversing a Linked List

(Display nodes in linked list / Count no. of nodes / Reach the end of the list)

Step 1: [INITIALIZE] SET PTR = START

Step 2: Repeat Steps 3 and 4 while PTR != NULL

Step 3: Apply Process to PTR-> DATA

Step 4: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 5: EXIT

```
Step 1: [INITIALIZE] SET COUNT = 0
Step 2: [INITIALIZE] SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR != NULL
Step 4:              SET COUNT = COUNT + 1
Step 5:              SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 6: Write COUNT
Step 7: EXIT
```

Figure 6.9 Algorithm to print the number of nodes in a linked list

Searching a Node in the Linked List

linked list.

Consider the linked list shown in Fig. 6.11. If we have VAL = 4, then the flow of the algorithm can be explained as shown in the figure.

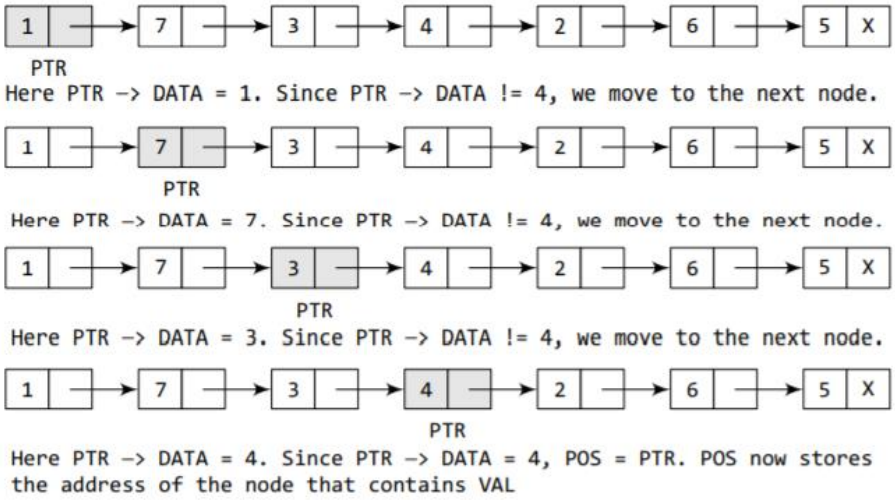


Figure 6.11 Searching a linked list

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Step 3 while PTR != NULL
Step 3:     IF VAL = PTR -> DATA
                SET POS = PTR
                Go To Step 5
            ELSE
                SET PTR = PTR -> NEXT
            [END OF IF]
        [END OF LOOP]
Step 4: SET POS = NULL
Step 5: EXIT
```

Figure 6.10 Algorithm to search a linked list

Inserting a Node at the Beginning of a Linked List

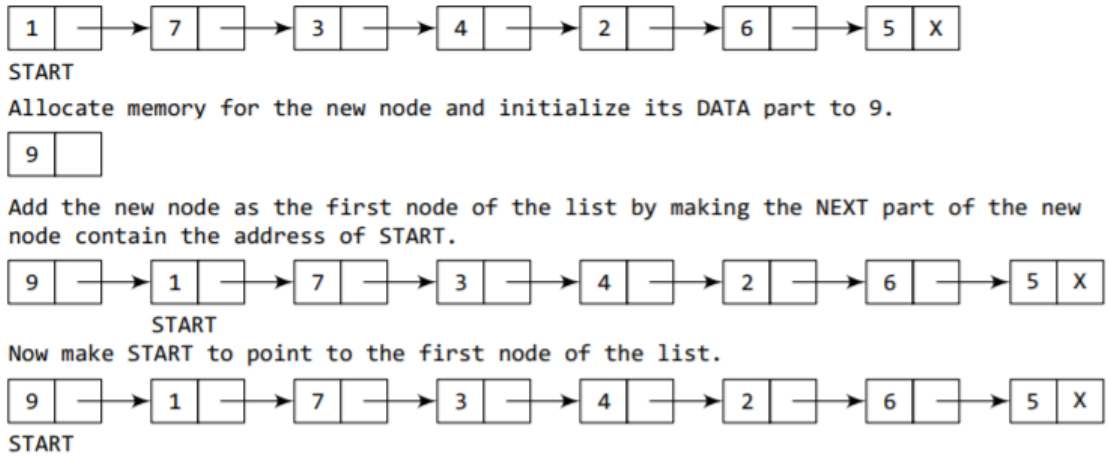


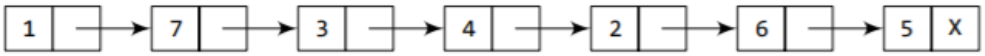
Figure 6.12 Inserting an element at the beginning of a linked list

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 7
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE -> DATA = VAL
Step 5: SET NEW_NODE -> NEXT = START
Step 6: SET START = NEW_NODE
Step 7: EXIT
```

Figure 6.13 Algorithm to insert a new node at the beginning

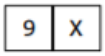
```
struct node *insert_beg(struct node *start)
{
        struct node *new_node;
        int num;
        printf("\n Enter the data : ");
        scanf("%d", &num);
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> data = num;
        new_node -> next = start;
        start = new_node;
        return start;
}
```

Inserting a Node at the End of a Linked List

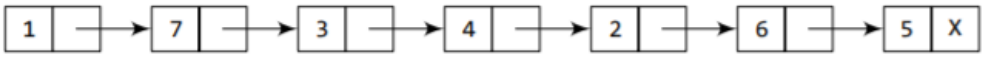


START

Allocate memory for the new node and initialize its DATA part to 9 and NEXT part to NULL.

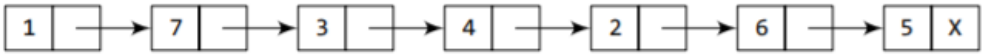


Take a pointer variable PTR which points to START.



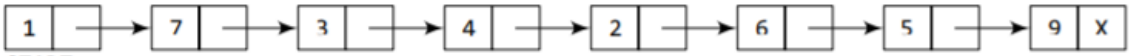
START, PTR

Move PTR so that it points to the last node of the list.



START PTR

Add the new node after the node pointed by PTR. This is done by storing the address of the new node in the NEXT part of PTR.



START PTR

Figure 6.14 Inserting an element at the end of a linked list

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 10
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET NEW_NODE->DATA = VAL
Step 5: SET NEW_NODE->NEXT = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR->NEXT != NULL
Step 8:     SET PTR = PTR->NEXT
        [END OF LOOP]
Step 9: SET PTR->NEXT = NEW_NODE
Step 10: EXIT
```

Figure 6.15 Algorithm to insert a new node at the end

Inserting a Node After a Given Node in a Linked List

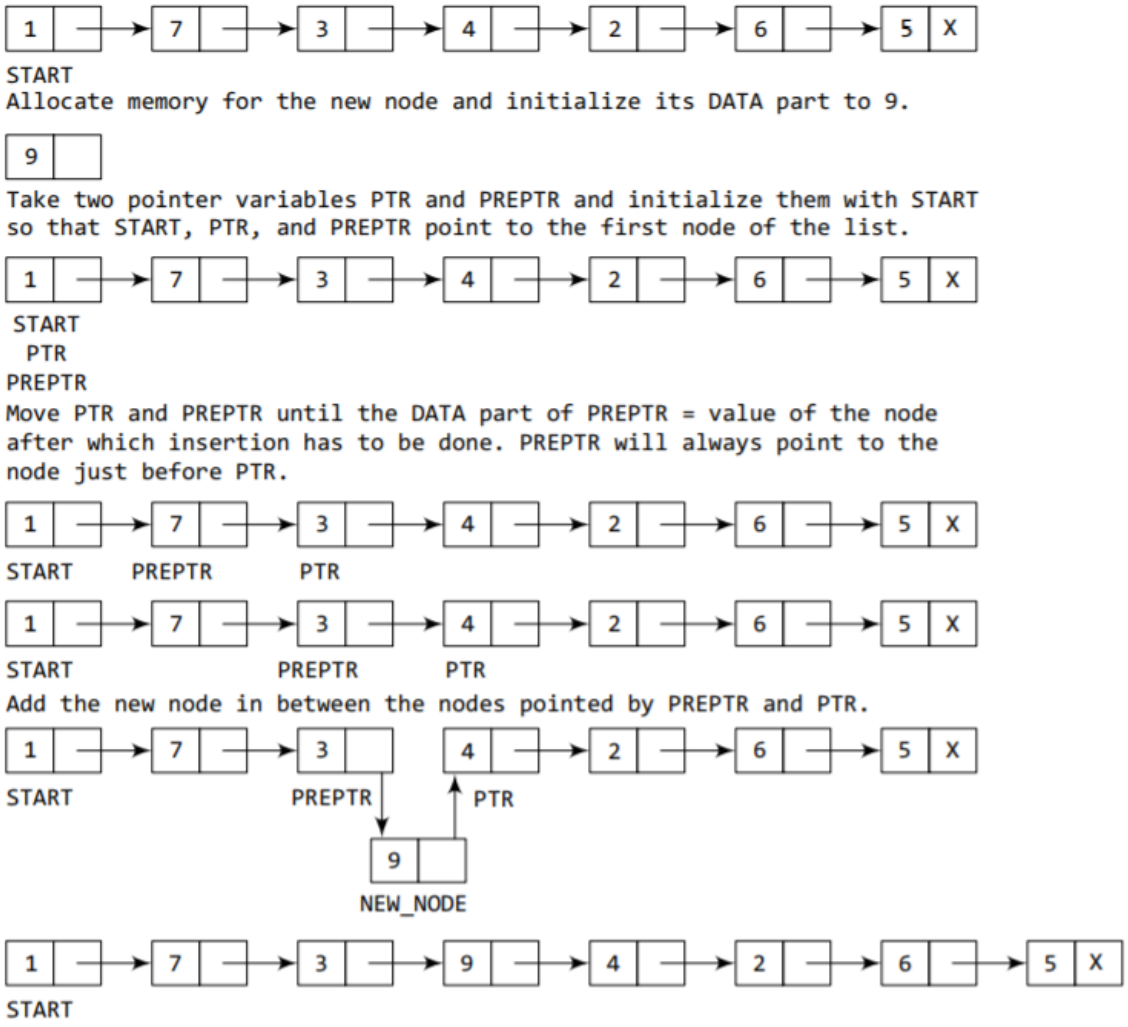


Figure 6.17 Inserting an element after a given node in a linked list

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 12
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET NEW_NODE->DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PREPTR->DATA
        != NUM
Step 8:     SET PREPTR = PTR
Step 9:     SET PTR = PTR->NEXT
         [END OF LOOP]
Step 10: PREPTR->NEXT = NEW_NODE
Step 11: SET NEW_NODE->NEXT = PTR
Step 12: EXIT
```

Figure 6.16 Algorithm to insert a new node after a node that has value NUM

Inserting a Node Before a Given Node in a Linked List

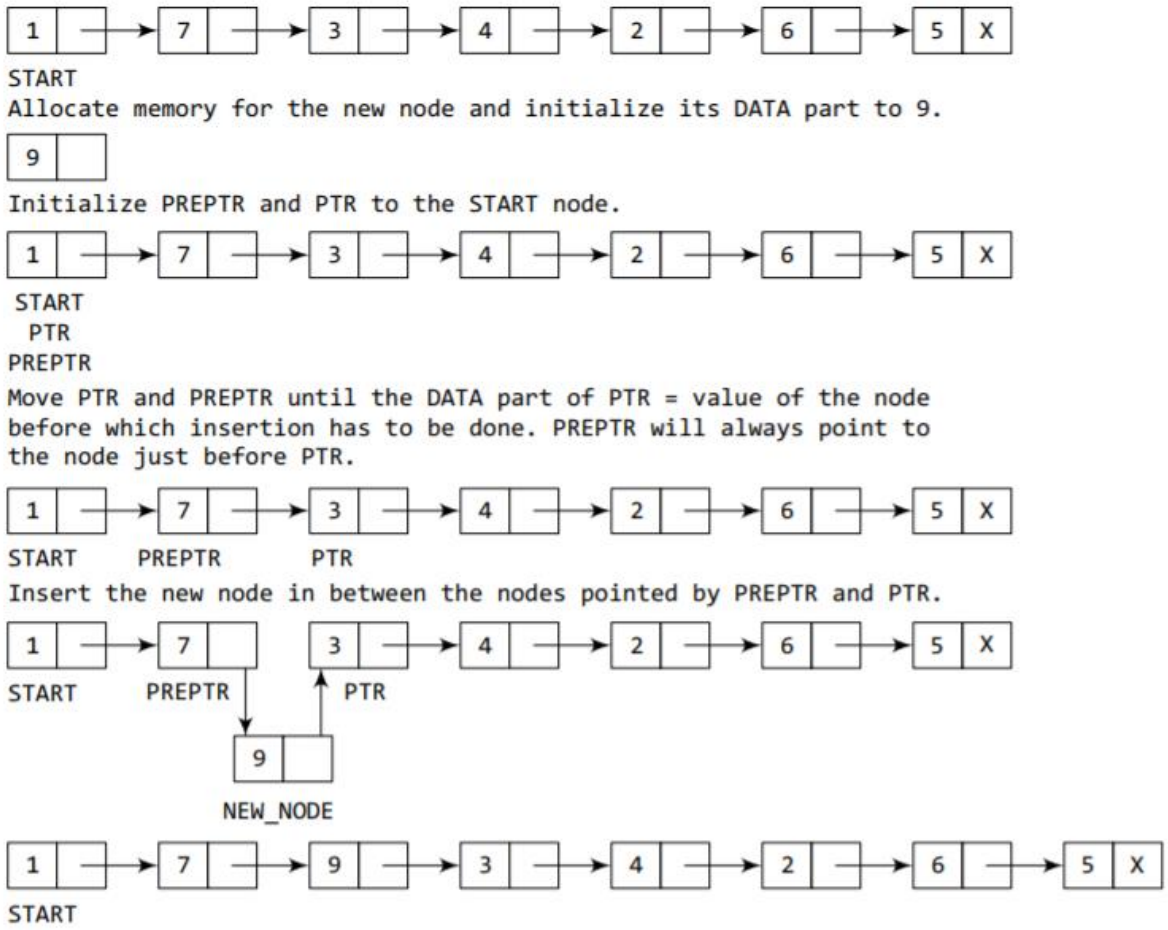


Figure 6.19 Inserting an element before a given node in a linked list

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 12
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE -> DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PTR -> DATA != NUM
Step 8:     SET PREPTR = PTR
Step 9:     SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 10: PREPTR -> NEXT = NEW_NODE
Step 11: SET NEW_NODE -> NEXT = PTR
Step 12: EXIT
```

Figure 6.18 Algorithm to insert a new node before a node that has value NUM

Deleting the First Node from a Linked List

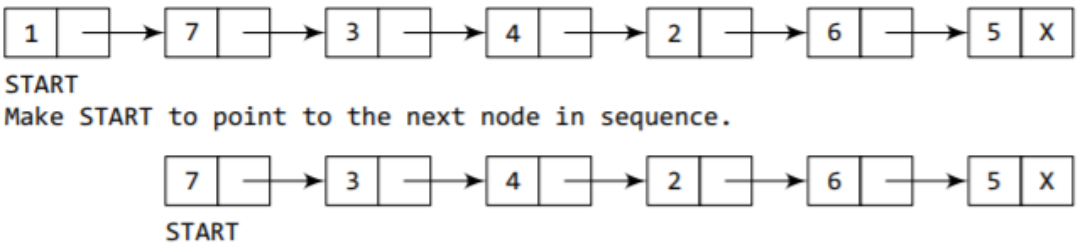


Figure 6.20 Deleting the first node of a linked list

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 5
        [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START -> NEXT
Step 4: FREE PTR
Step 5: EXIT
```

Figure 6.21 Algorithm to delete the first node

Deleting the Last Node from a Linked List

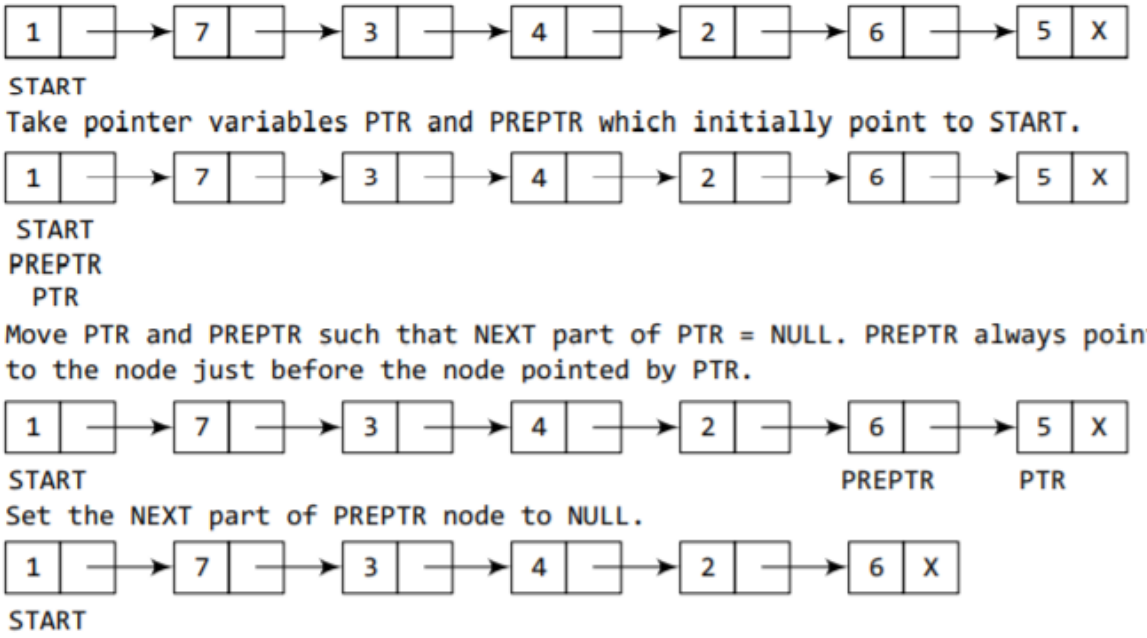


Figure 6.22 Deleting the last node of a linked list

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 8
        [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR -> NEXT != NULL
Step 4:     SET PREPTR = PTR
Step 5:     SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 6: SET PREPTR -> NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

Figure 6.23 Algorithm to delete the last node

Deleting the Node After a Given Node in a Linked List

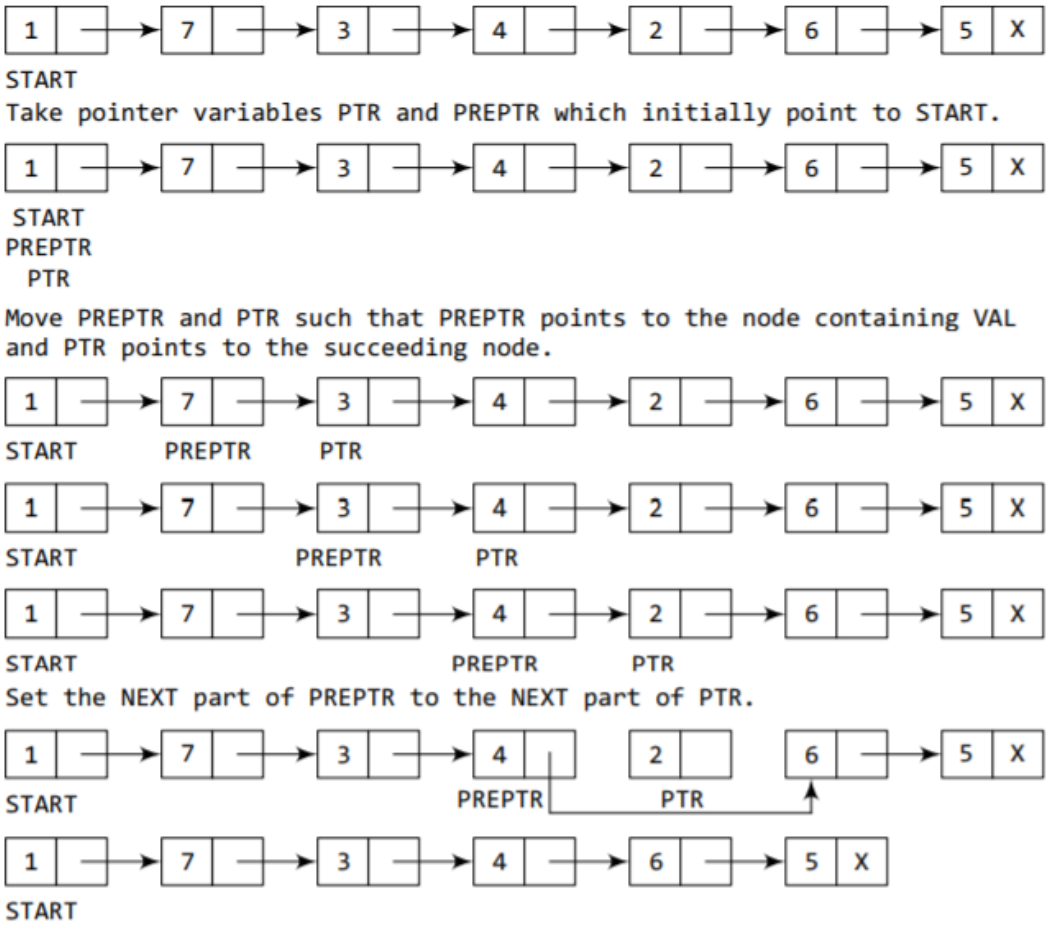


Figure 6.24 Deleting the node after a given node in a linked list

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 10
        [END OF IF]
Step 2: SET PTR = START
Step 3: SET PREPTR = PTR
Step 4: Repeat Steps 5 and 6 while PREPTR -> DATA != NUM
Step 5:     SET PREPTR = PTR
Step 6:     SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 7: SET TEMP = PTR
Step 8: SET PREPTR -> NEXT = PTR -> NEXT
Step 9: FREE TEMP
Step 10: EXIT
```

Figure 6.25 Algorithm to delete the node after a given node