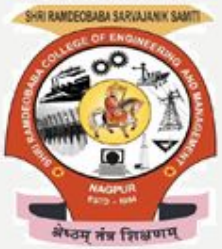


SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR - 440013

Data structures and algorithm
(CST252)
III semester section a

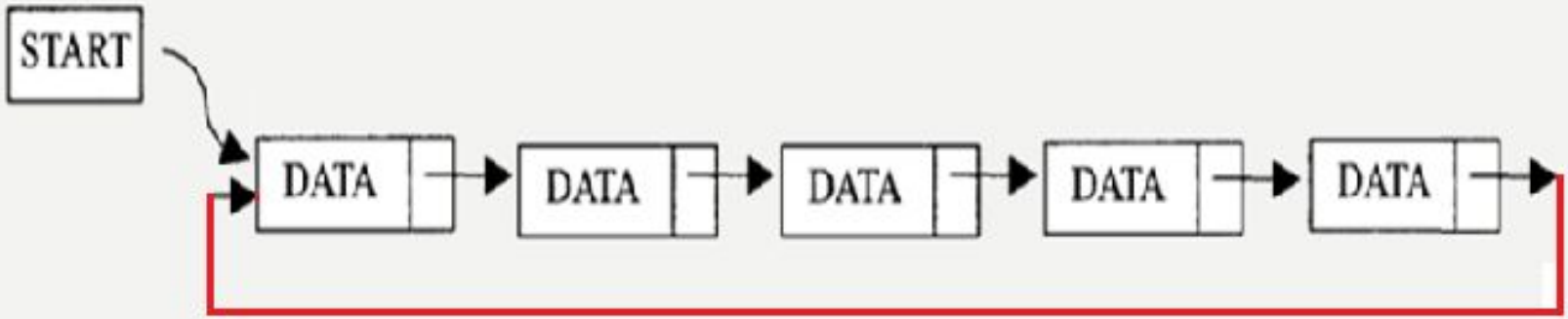
Course Coordinator: Rina Damdoo

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Circular LL

Circular singly Linked list

- Circular singly linked list is a linked list in which last node contains a link to first/start node



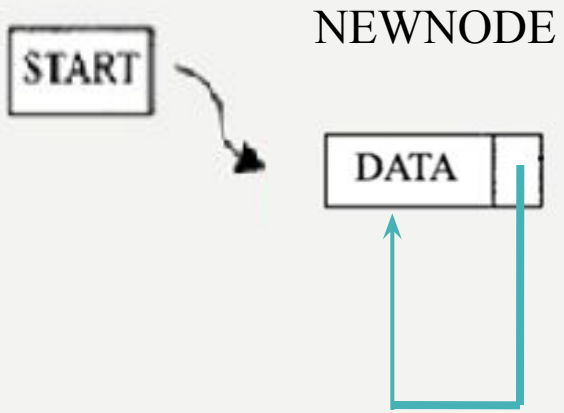
Circular singly Linked list

Node Declaration:

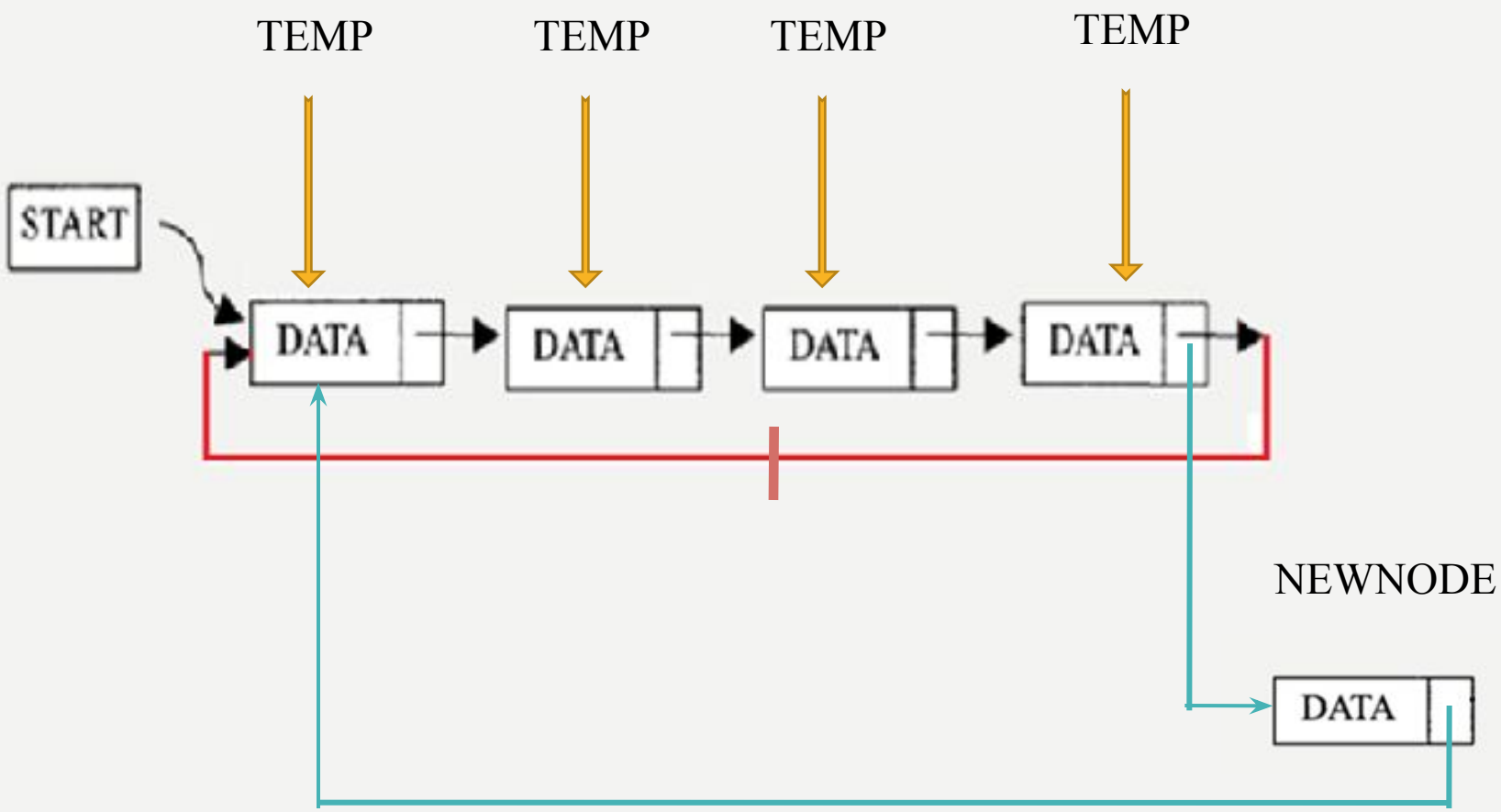
```
typedef struct Node
{
int data;
struct Node *next;
} node;


node *start=NULL;
```

Circular Linked list creation



Circular Linked list creation



temp -> next= Newnode;

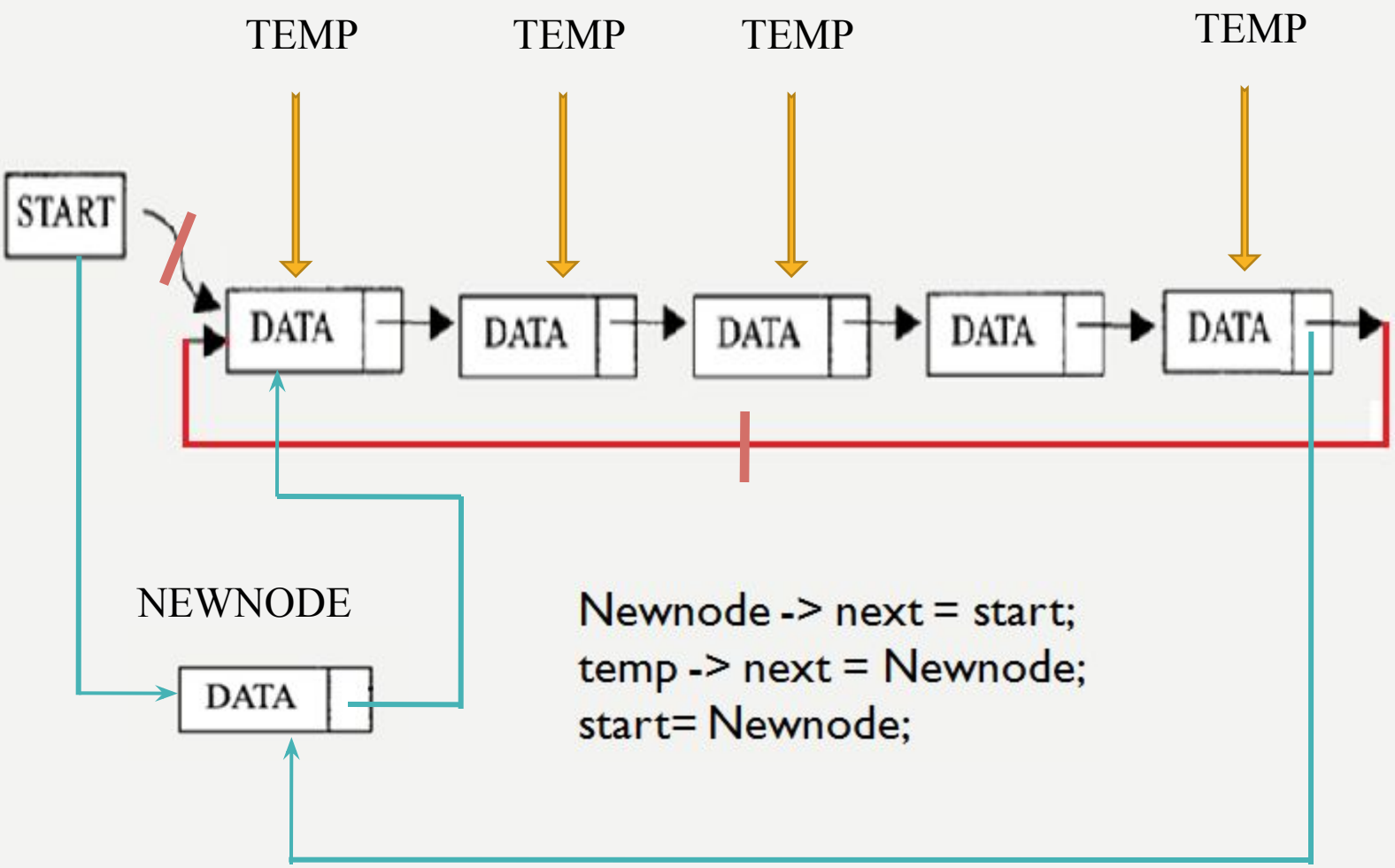
Newnode -> next=start;

Circular Linked list creation

```
Newnode=(node*)malloc(sizeof(node));
Newnode -> data=x;
Newnode -> next=NULL;
if(start == NULL)
{
     start = Newnode ;
     start -> next=start;
}
else
{
     temp=start;
     while(temp -> next != start)
          temp=temp -> next;
     temp -> next= Newnode;
     Newnode -> next=start;
}
```

insertion

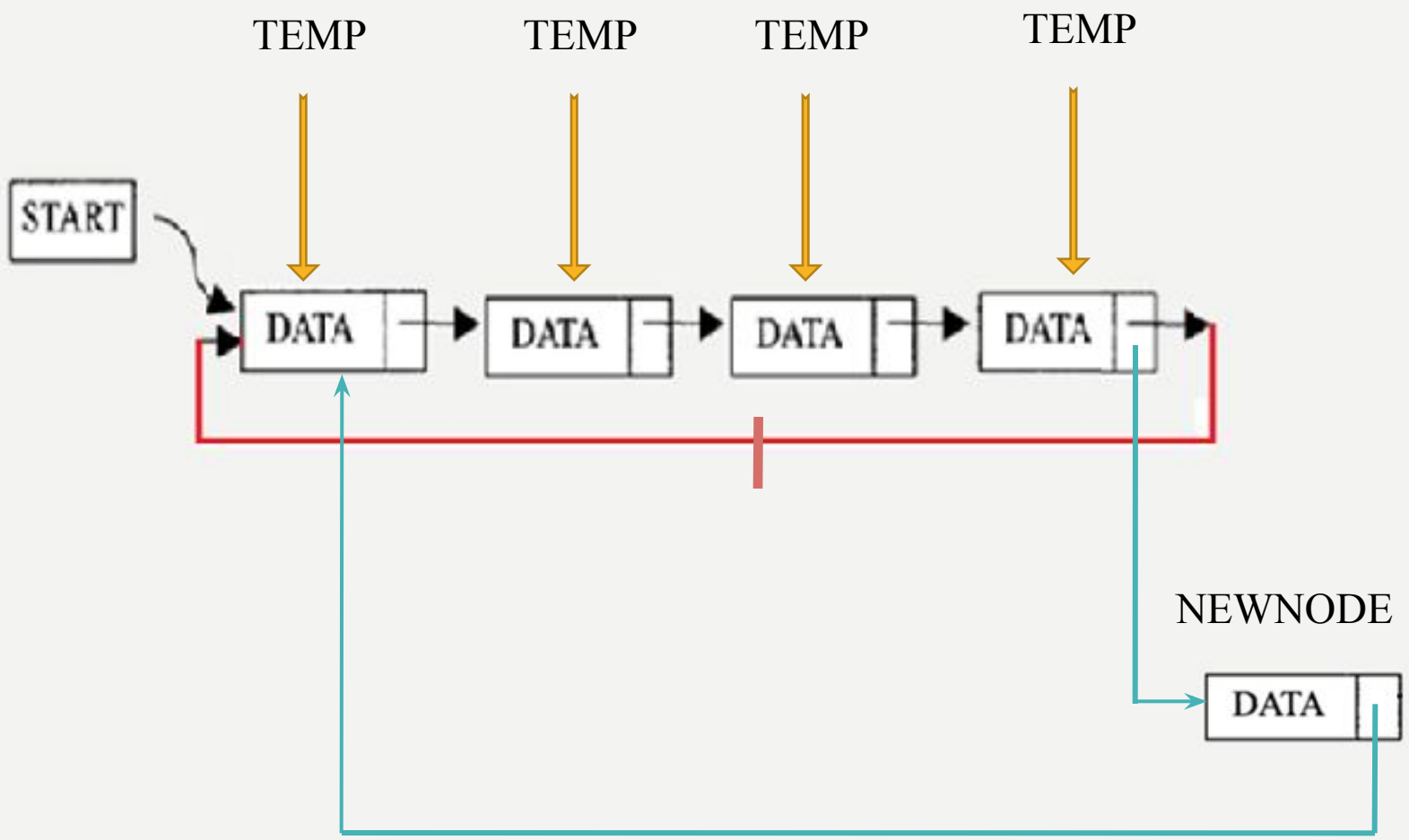
insertion as first node



insertion as first node

```
temp=start;
while(temp -> next != start)
    temp= temp -> next;
Newnode -> next = start;
temp -> next = Newnode;
start= Newnode;
```

insertion as last node

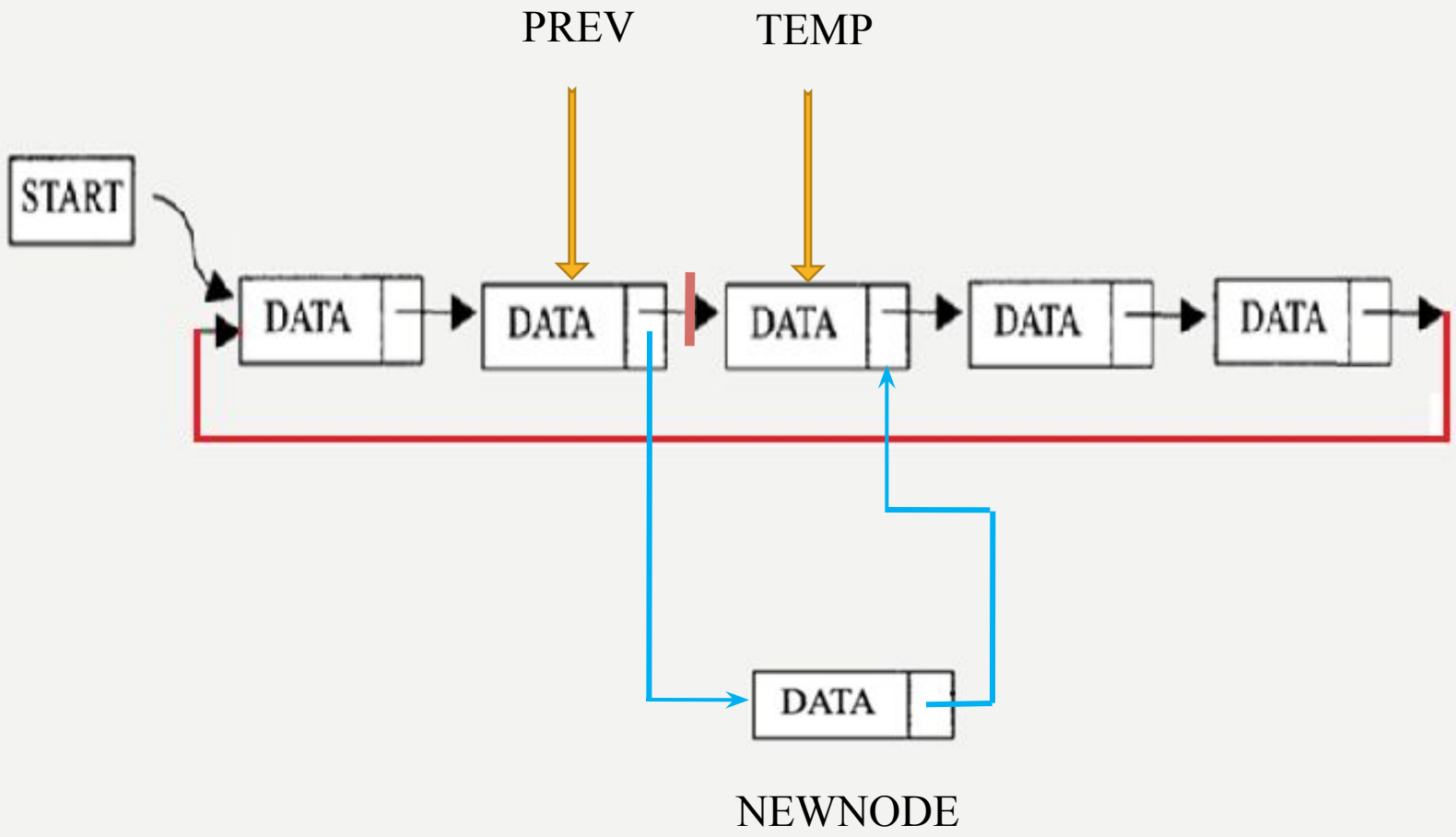



```
temp -> next = Newnode;
Newnode -> next = start;
```

insertion as last node

```
temp=start;
while(temp -> next != start)
    temp= temp -> next;
temp -> next = Newnode;
Newnode -> next = start;
```

Insertion at the specified position



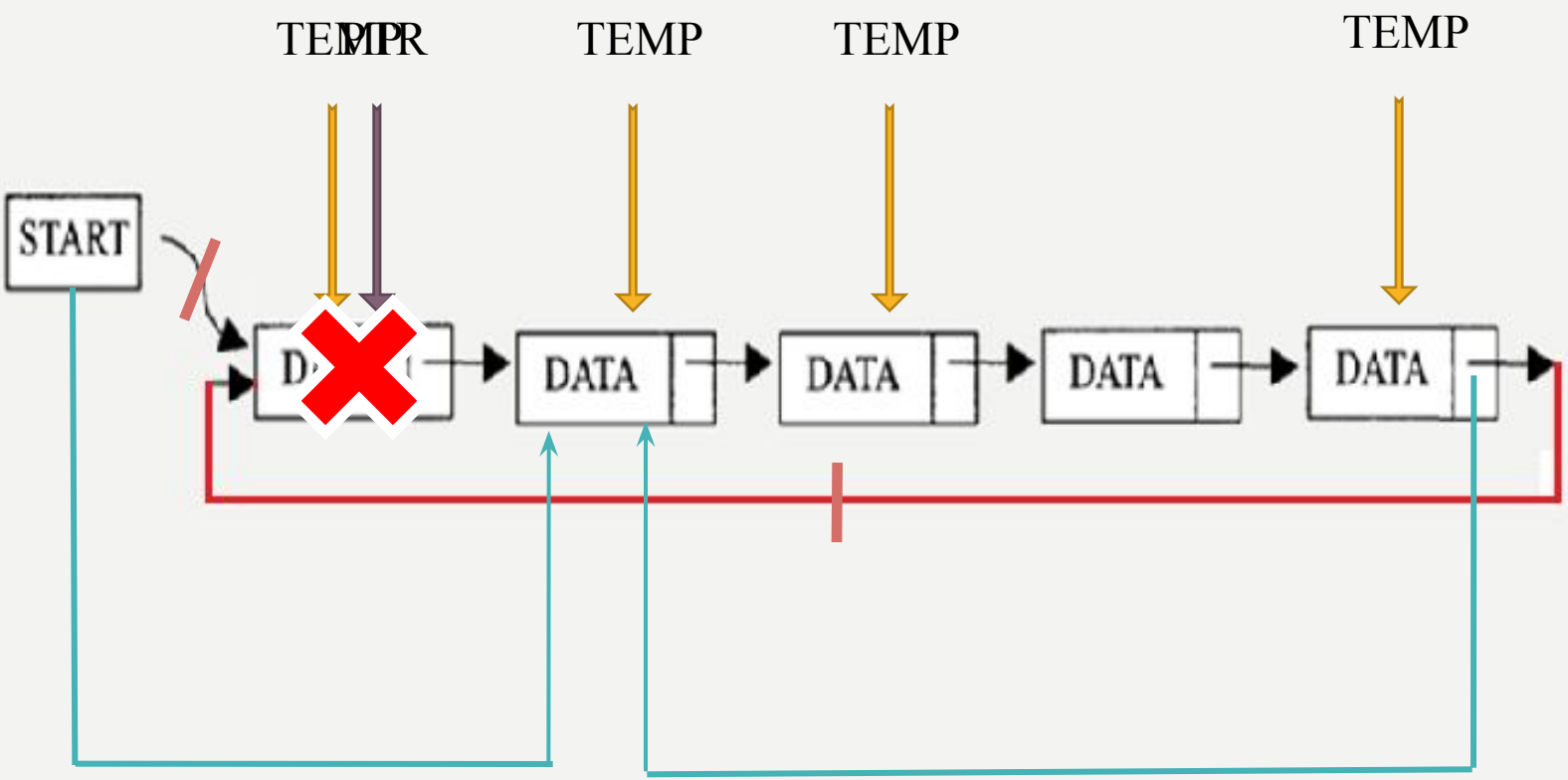

```
Newnode -> next =temp;
prev -> next = Nwenode;
```

Insertion at the specified position

```
temp = start;
do
{
    prev = temp;
    temp = temp -> next;
    i++;
} while(temp != start && i != pos);
if(temp == start)
    printf("No sufficient number of nodes");
else
{
    Newnode -> next =temp;
    prev -> next = Newnode;
}
```

DELETION

deletion of first node

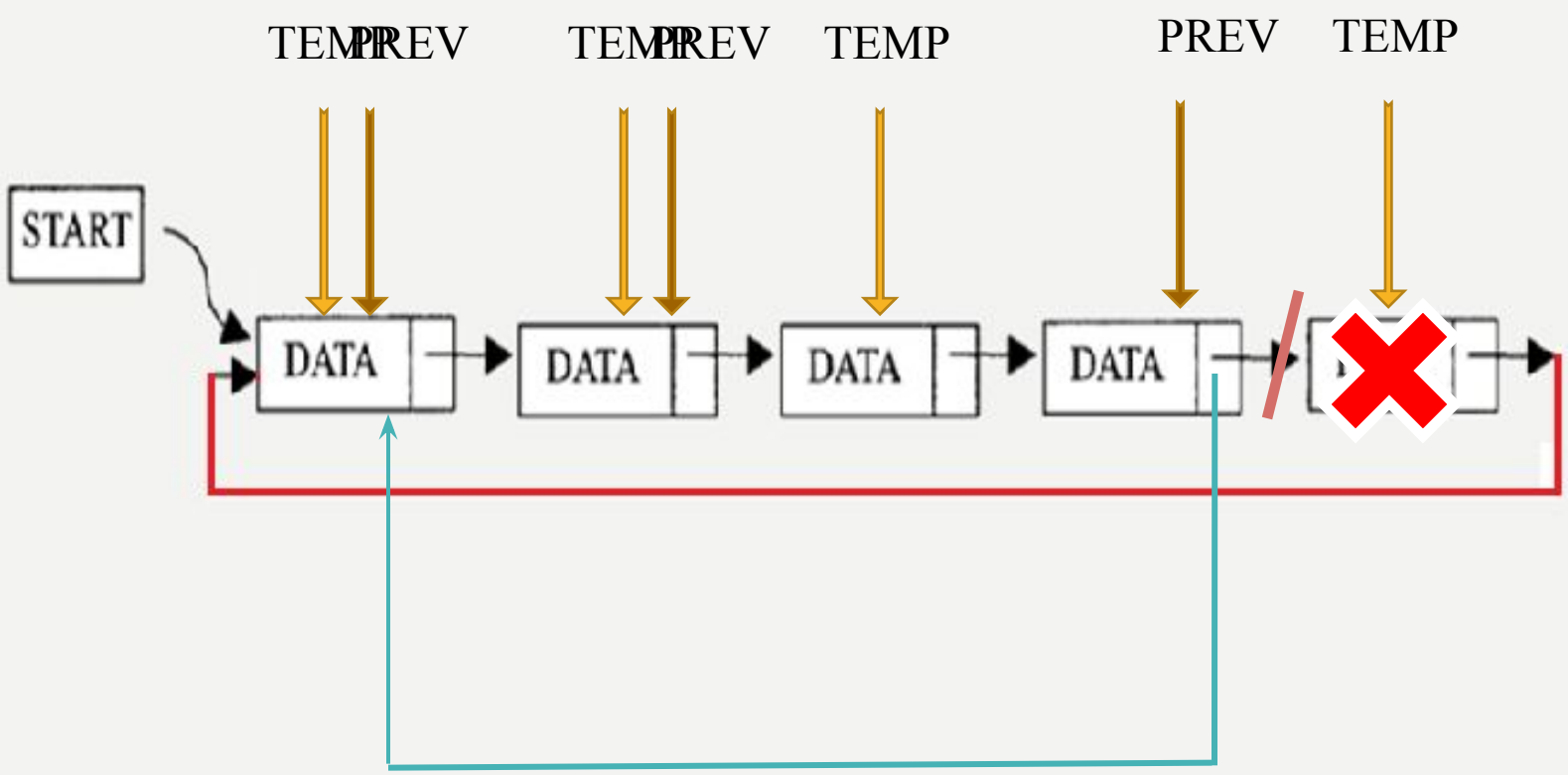



```
ptr= start;
start = start ->next;
temp ->next = start;
free(ptr);
```

DELETION OF THE first NODE

```
temp = start;
while(temp -> next != start )
      temp = temp -> next;
ptr= start;
start = start ->next;
temp ->next = start;
free(ptr);
```

DELETION OF THE last NODE

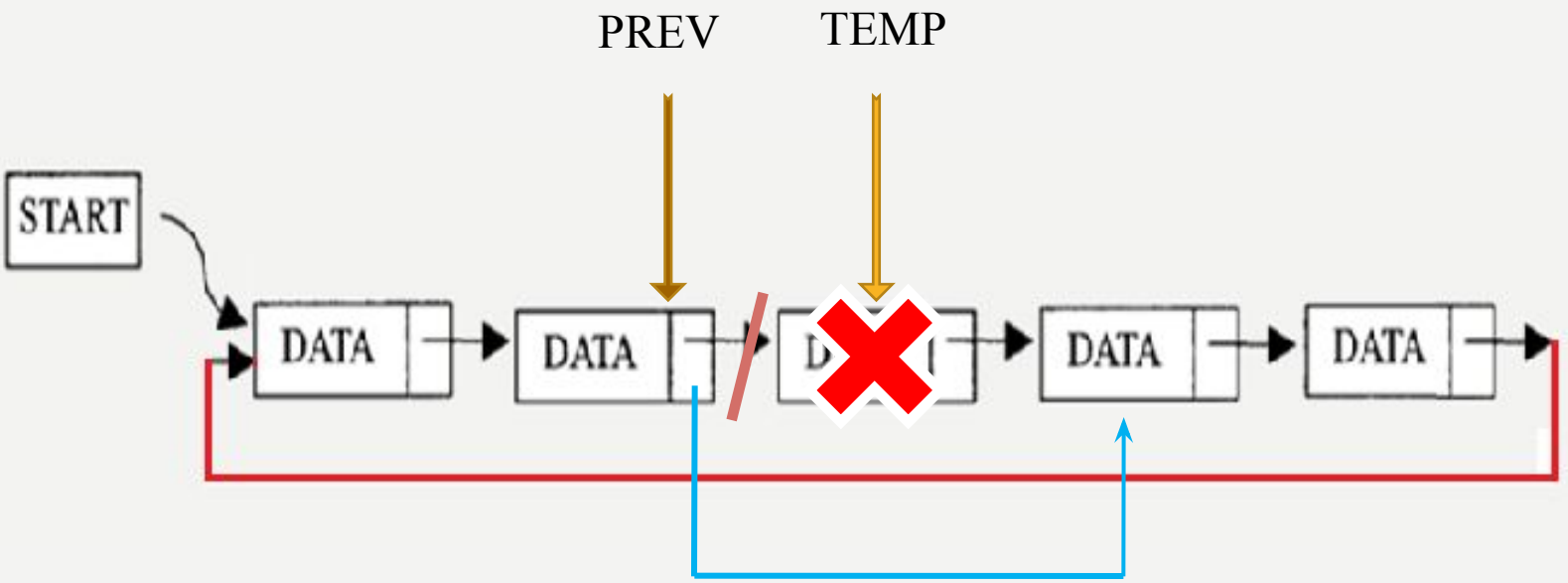



```
prev -> next = start;
free( temp);
```

DELETION OF THE last NODE

```
temp = start;
while(temp -> next != start )
{
    prev = temp;
    temp = temp -> next;
}
prev -> next = start;
free( temp);
```

DELETION OF THE NODE in between




```
prev -> next=temp-> next;
free(temp);
```

DELETION OF THE NODE in between

```
temp = start;
do
{
    prev = temp;
    temp = temp -> next;
    i++;
} while(temp != start && i != pos);
if(temp == start)
    printf("No sufficient number of nodes");
else
{
    prev -> next=temp-> next;
    free(temp);
}
```

Update operation on circular linked list

```
temp=start;
do
{
    if(temp->data==x)
        break;
    else
        temp=temp->next;
} while(temp!=start);
if ( temp==start)
    printf("data to be updated is not present");
else
    temp->data=xnew;
```

Display operation on circular linked list

```
if(start==NULL)
    printf("*********List is empty********");
else
{
    temp=start;
    do
    {
        printf("\t%d", temp->data);
        temp=temp->next;
    } while(temp != start);
}
```

Applications of circular linked list

- An application where any node can be a starting point, we can traverse the whole list by starting from any node and just need to stop when the first visited node is visited again.

- Circular lists are useful in applications to repeatedly go around the list. For example Round Robin (RR) job scheduling by operating system