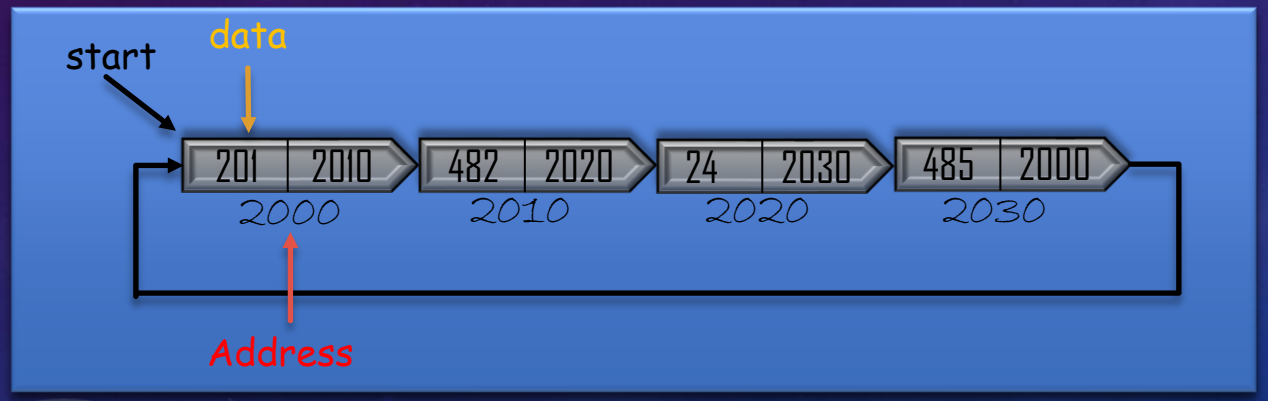
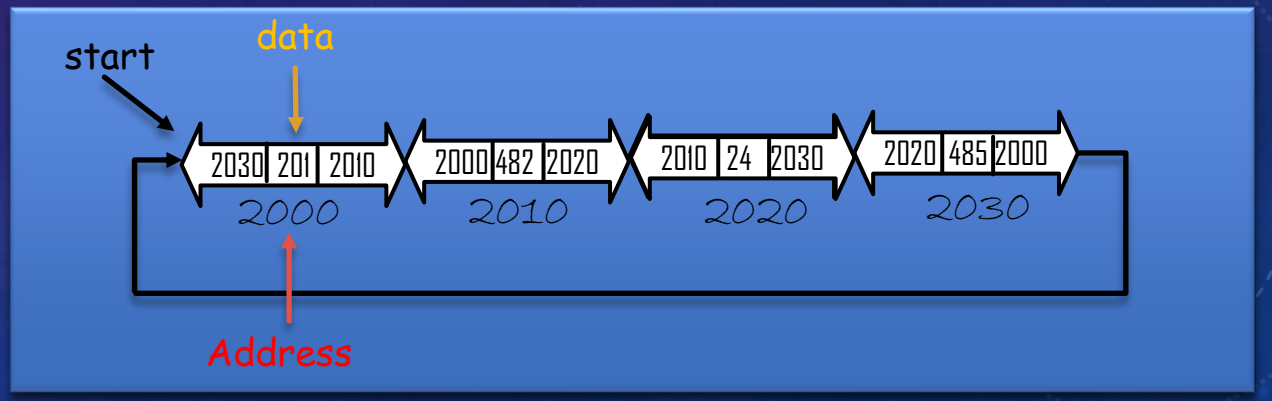
Circular Linked list

Made By: Yagneswara Naidu (CSE)

DEFINITION

Circular Linked List: It is a variation of linked list in which last node points to first node's address. Both single and double linked list can be drawn for circular linked list.





SINGLE CIRCULAR LINKED LIST

Insertion at Beginning

Step 1: START

Step 2: Create a new node i.e. “temp”

Step 3: Read data “x”

```
temp->data=x;
temp->next=NULL;
```

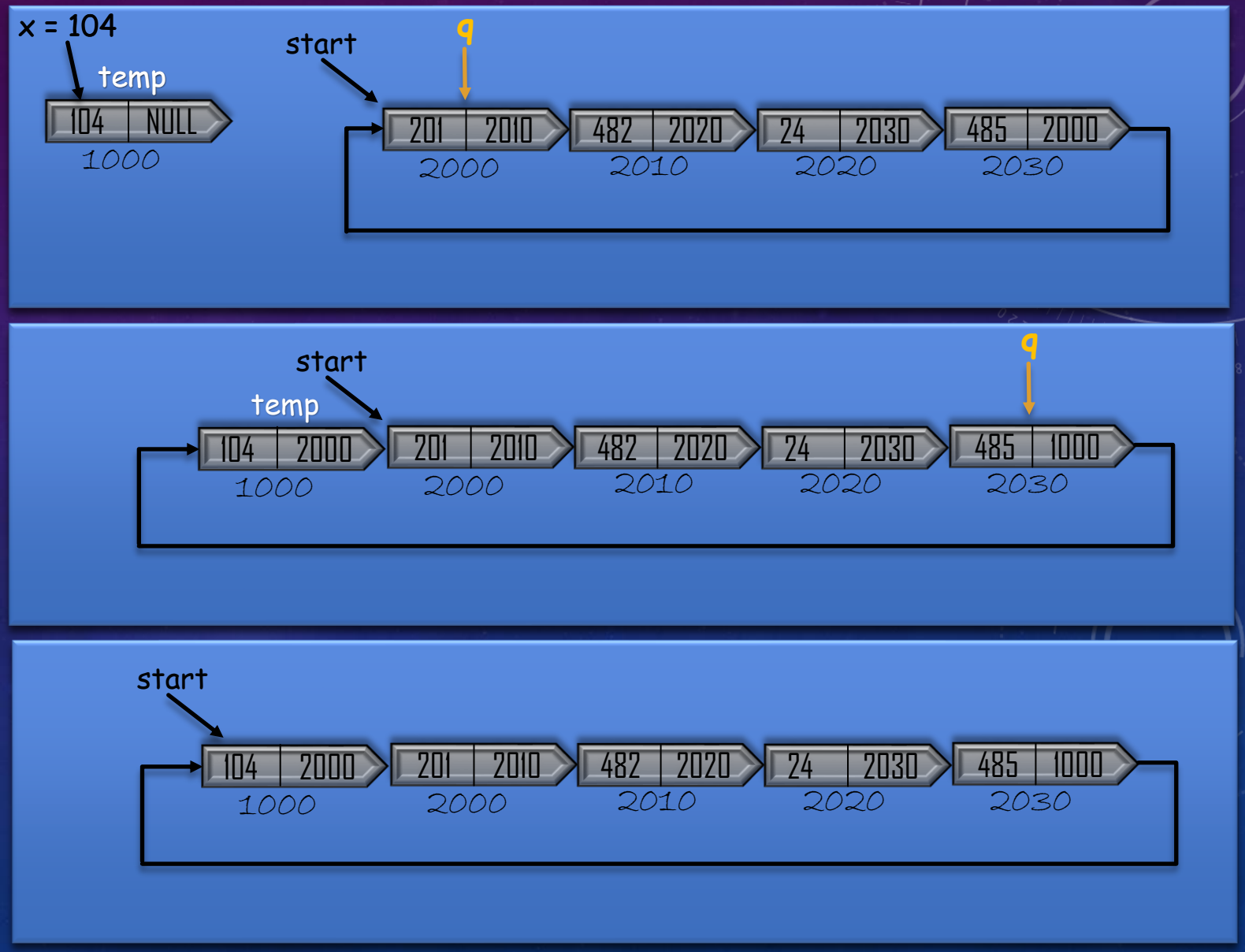
Step 4: q=start

```
while(q->next!=start)
{
    q=q->next;
}
```

Step 5: q->next=temp;

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

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Insertion at Position/Middle

Step 1: START

Step 2: Create a new node i.e. "temp"

Step 3: Read data "x"

```
temp->data=x;
temp->next=NULL;
```

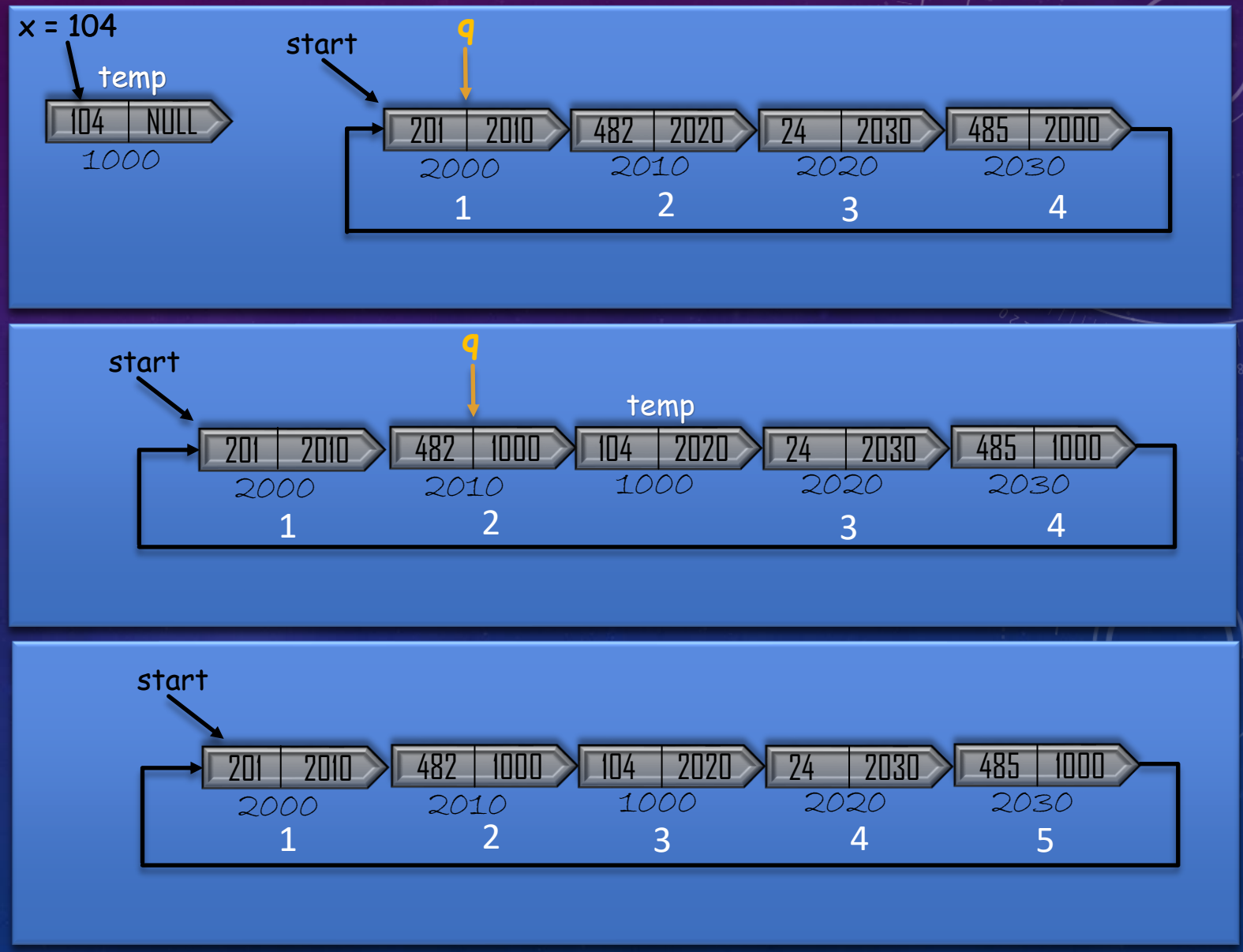
Step 4: Read position i.e. "pos"

```
q=start;
for(i=1;i<pos;i++)
{
    q=q->next;
}
```

Step 5:

```
temp->next=q->next;
q->next=temp;
```

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Insertion at End

Step 1: START

Step 2: Create a new node i.e. “temp”

Step 3: Read data “x”

```
temp->data=x;
temp->next=NULL;
```

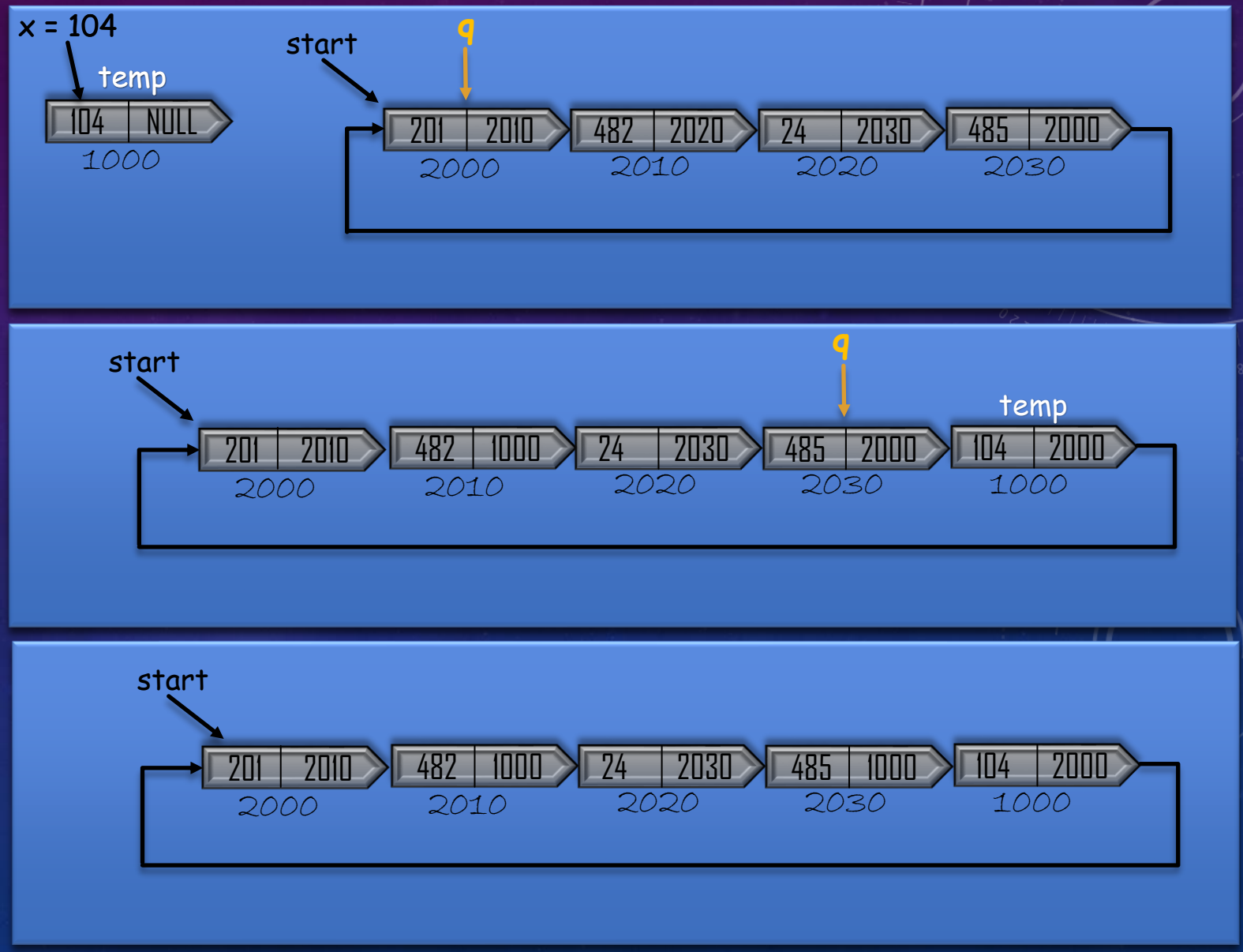
Step 4: q=start

```
while(q->next!=start)
{
    q=q->next;
}
```

Step 5: q->next=temp;

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

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Deletion from Beginning

Step 1: START

Step 2: temp=start

Step 3: q=start

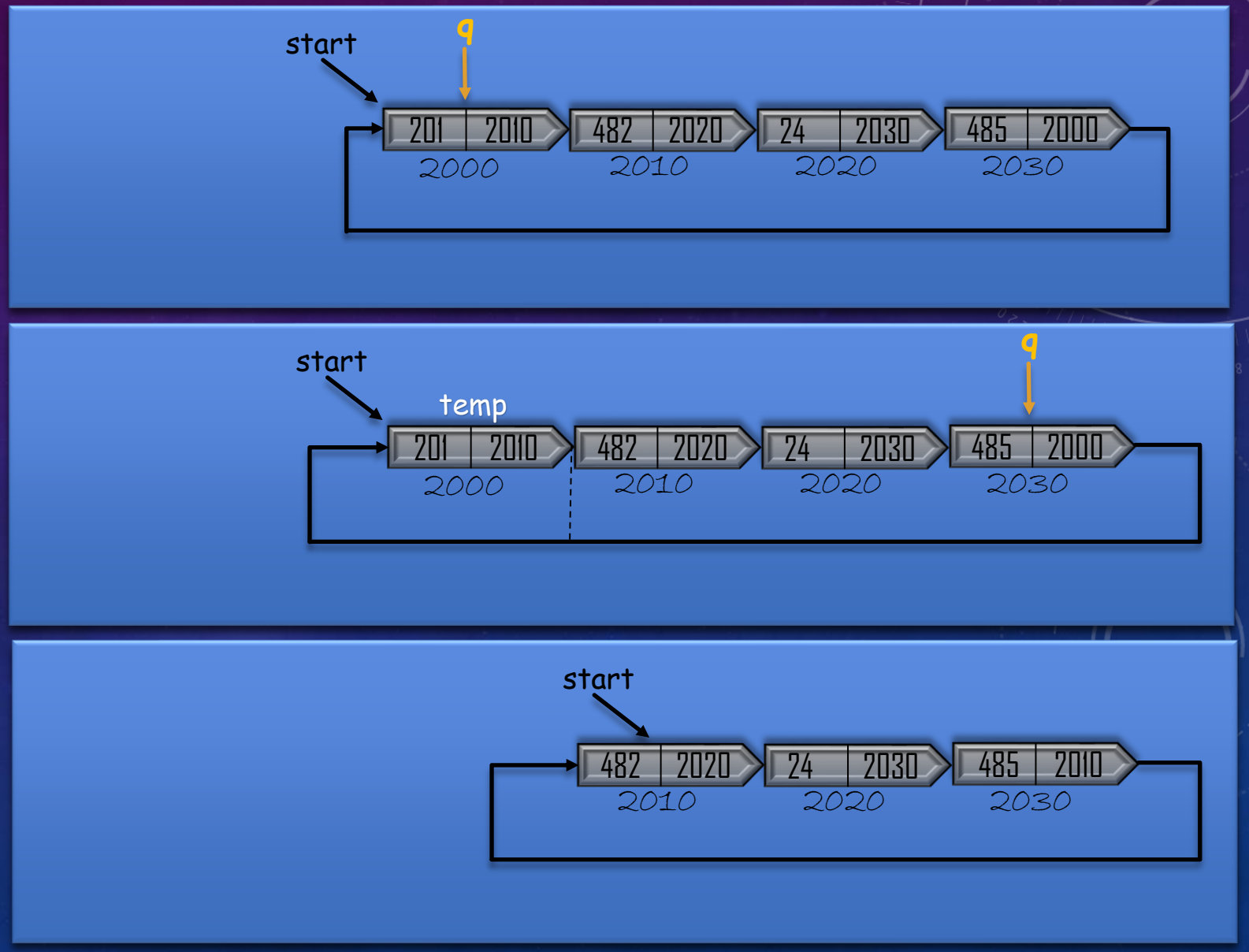
```
while(q->next!=start)
{
    q=q->next;
}
```

Step 4: q->next=start->next;

start=start->next;

Step 5: free(temp);

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Deletion from Position/Middle

Step 1: START

Step 2: Read position i.e. “pos”

Step 3: q=start;

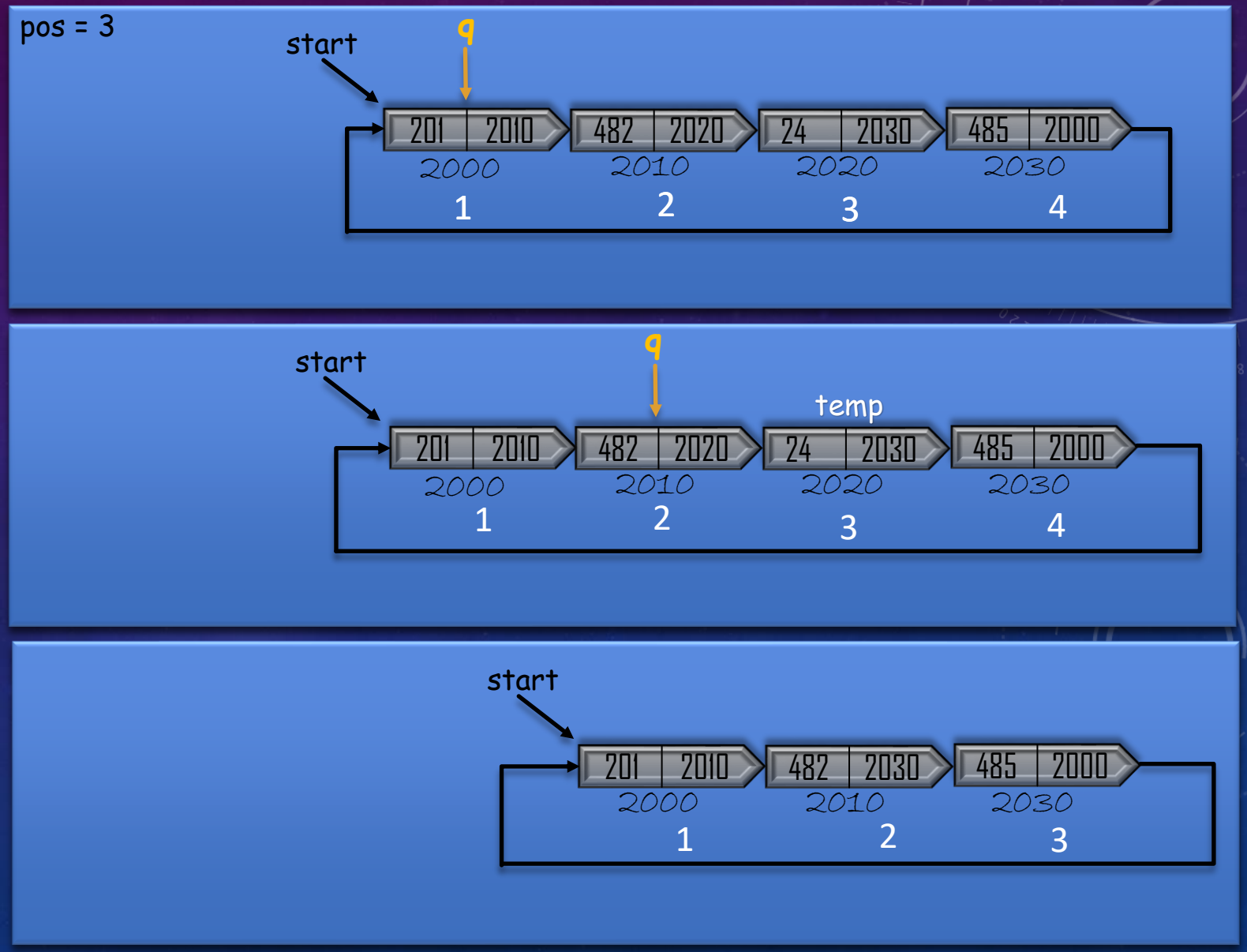
```
for(i=1;i<pos;i++)
{
    q=q->next;
}
```

Step 4: temp=q->next;

```
q->next=temp->next;
```

Step 5: free(temp);

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Deletion from End

Step 1: START

Step 2: q=start

Step 3: while(q->next->next!=start)

{

q=q->next;

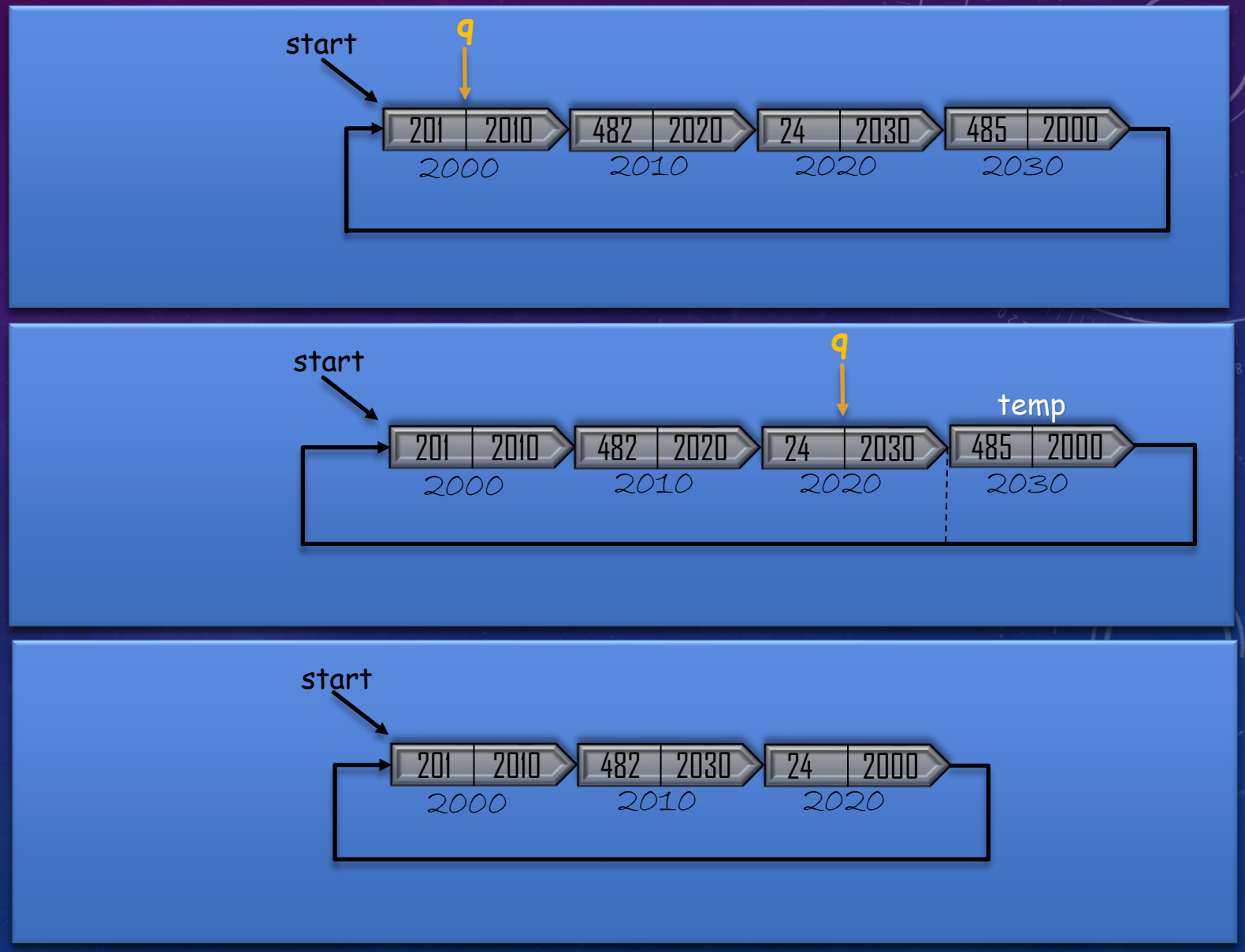
}

Step 4: temp=q->next;

q->next=start;

Step 5: free(temp);

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Display Operation

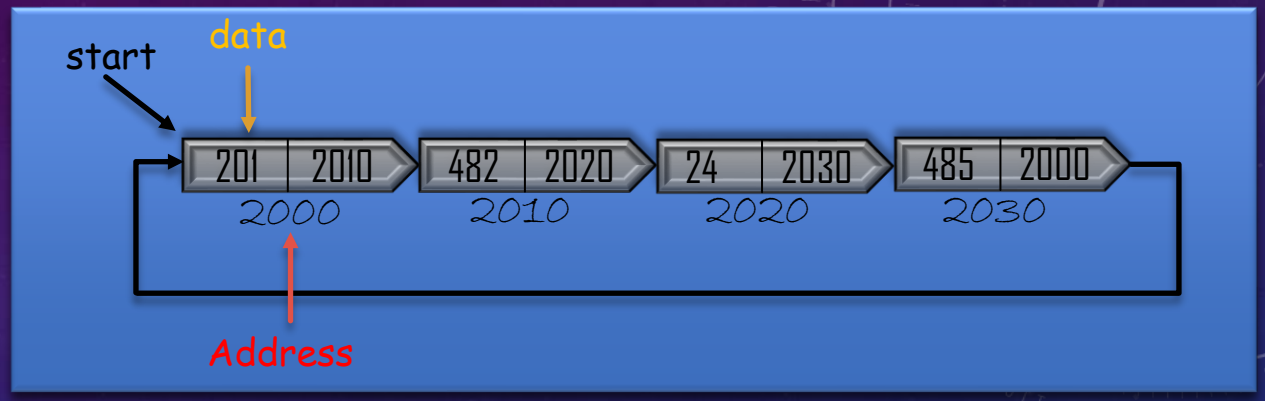
Step 1: START

Step 2: q=start

Step 3: while(q->next!=start)

```
{
    cout<<q->data<<" -> ";
    q=q->next;
}
```

Step 4: STOP



DOUBLE CIRCULAR LINKED LIST

Insertion at Beginning

Step 1: START

Step 2: Create a new node i.e. “temp”

Step 3: Read data “x”

temp->data=x;

temp->next=NULL;

temp->prev=NULL;

Step 4: start->prev->next=temp;

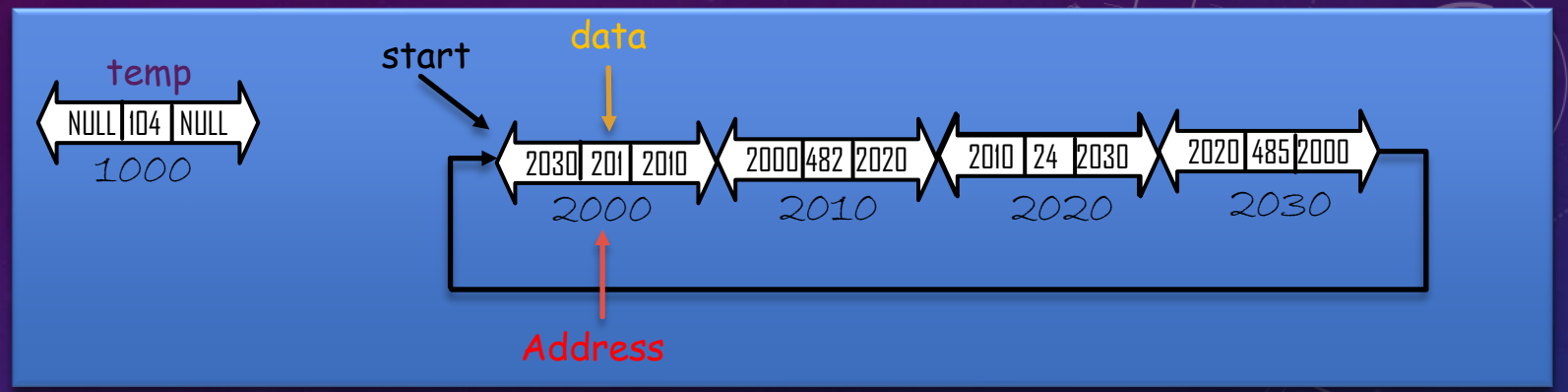
temp->prev=start->prev;

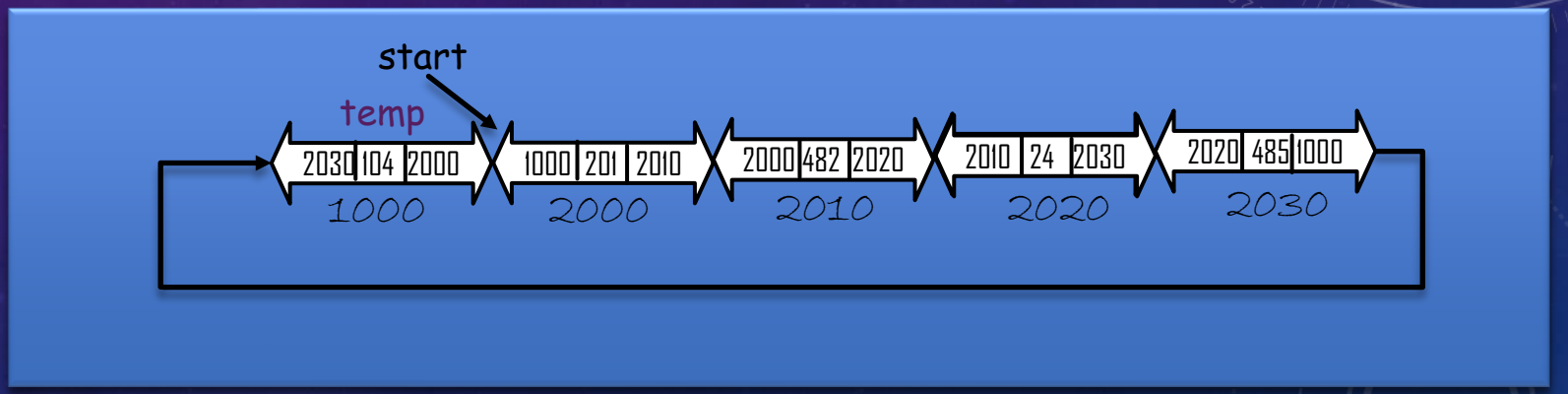
temp->next=start;

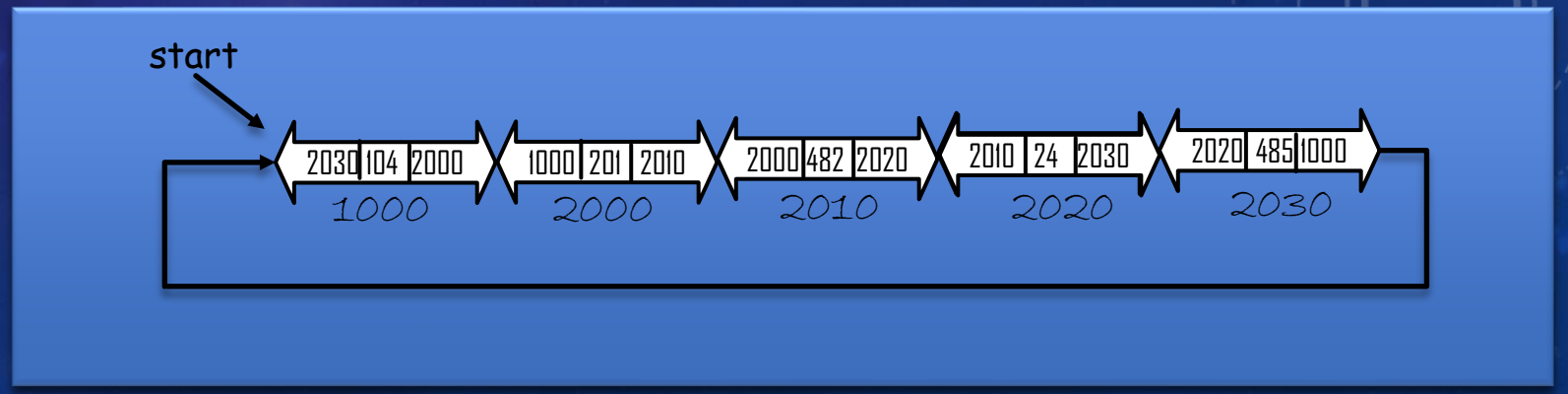
start->prev=temp;

Step 5: start=temp;

Step 6: STOP







Made By: Yagneswara Naidu (CSE)

DOUBLE CIRCULAR LINKED LIST

Insertion at Position/Middle

Step 1: START

Step 2: Create a new node i.e. "temp"

Step 3: Read data "x"

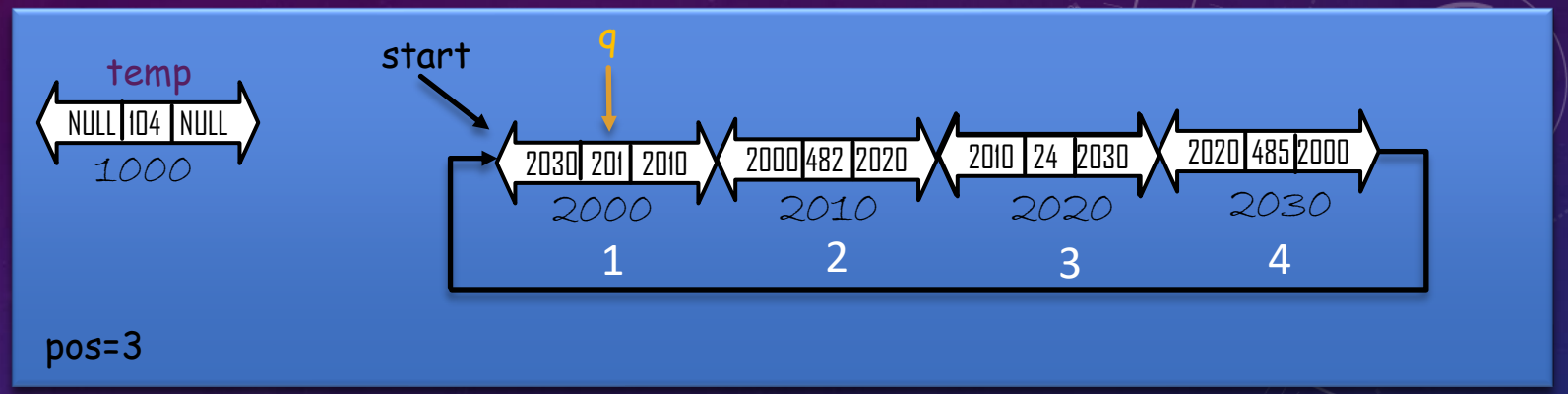
```
temp->data=x;
temp->next=NULL;
temp->prev=NULL;
```

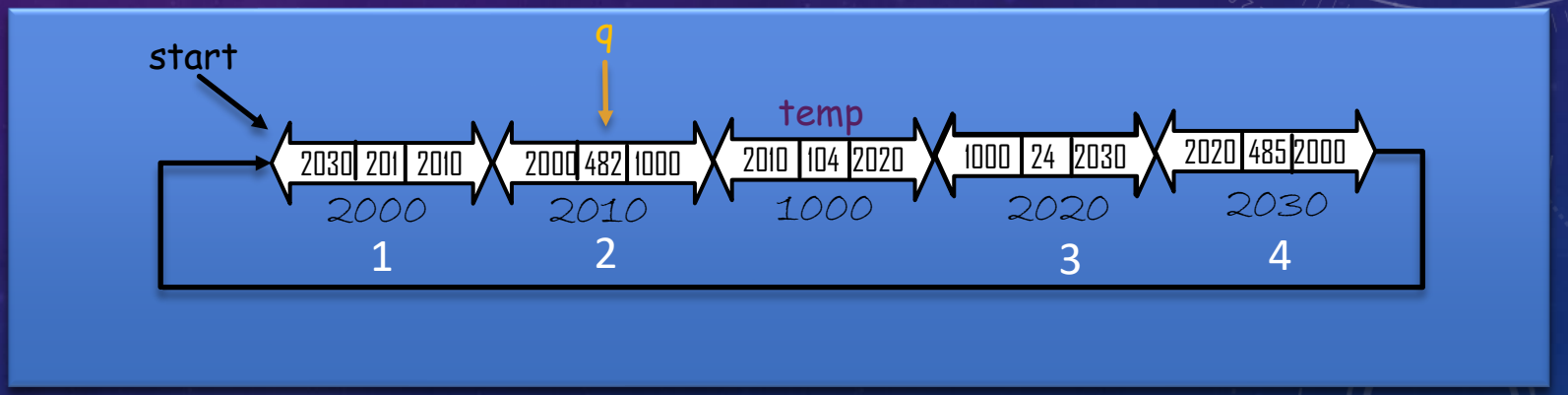
Step 4: Read position i.e. "pos"

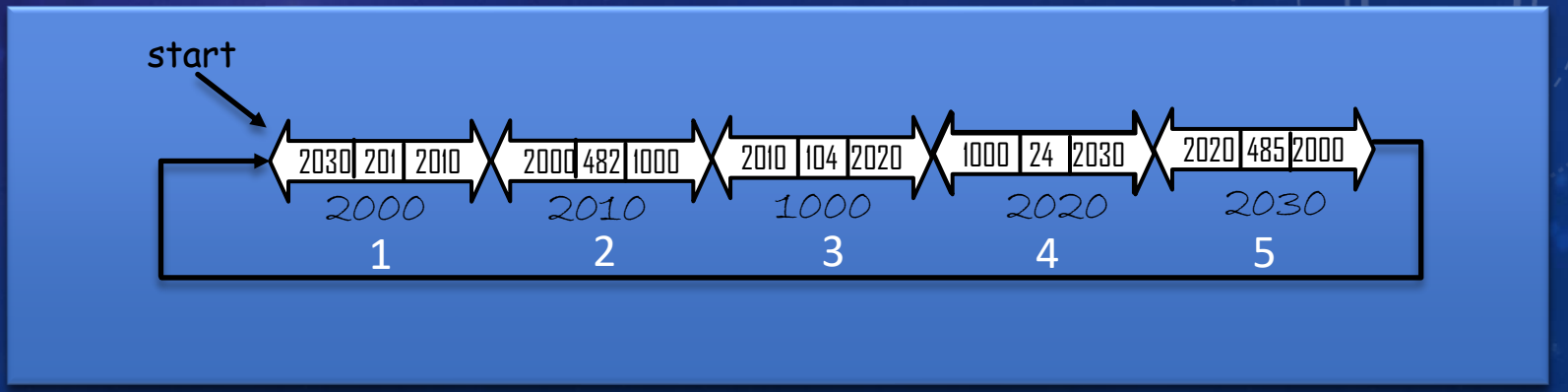
```
q=start;
for(i=1;i<pos;i++)
{
        q=q->next;
}
q->next->prev=temp;
temp->next=q->next;
temp->prev=q;
```

Step 5: q->next=temp;

Step 6: STOP







DOUBLE CIRCULAR LINKED LIST

Insertion at End

Step 1: START

Step 2: Create a new node i.e. “temp”

Step 3: Read data “x”

```
temp->data=x;
temp->next=NULL;
temp->prev=NULL;
```

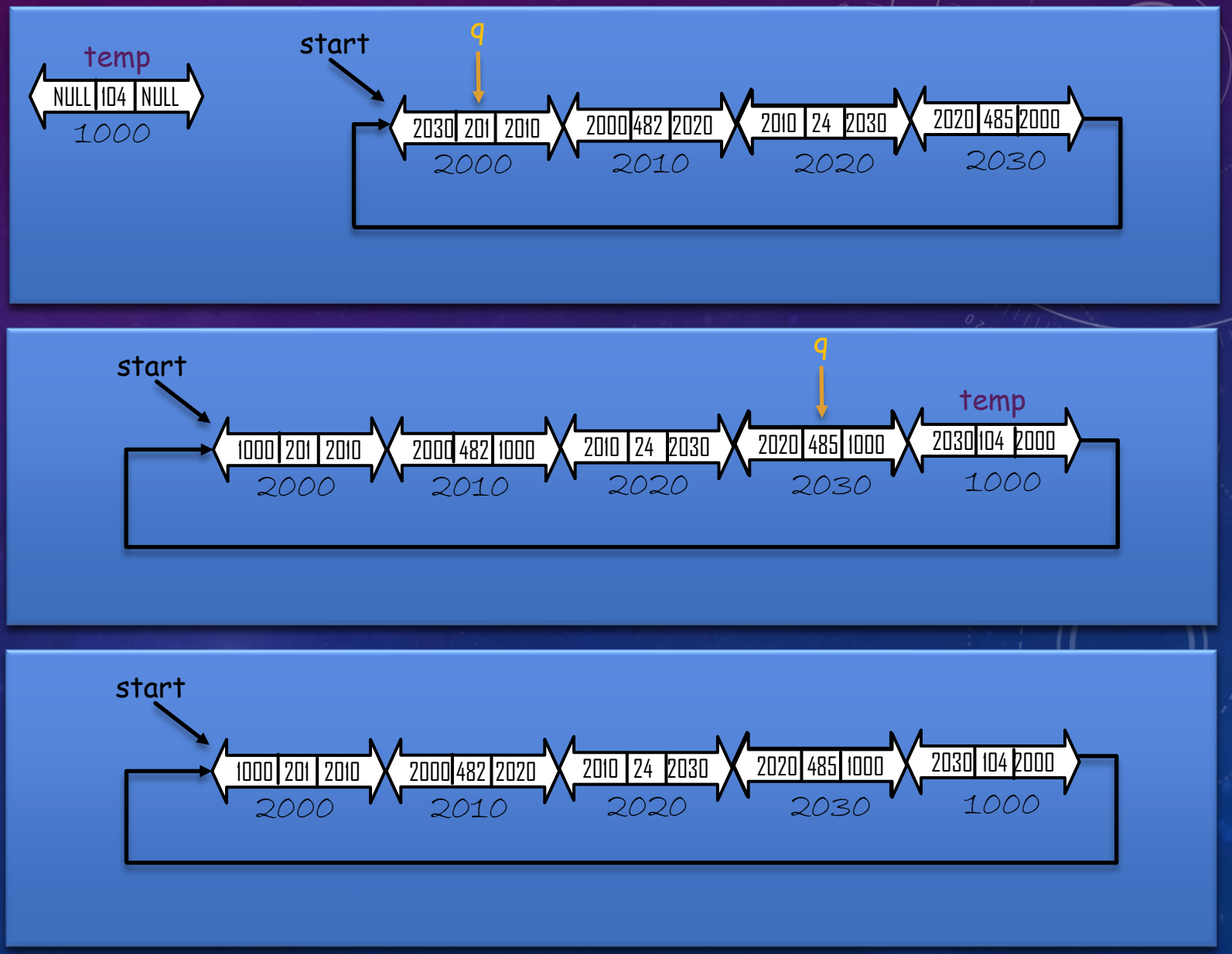
Step 4:

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

Step 5:

```
temp->prev=q;
q->next=temp;
```

Step 6: STOP



DOUBLE CIRCULAR LINKED LIST

Deletion from Beginning

Step 1: START

Step 2: temp=start;

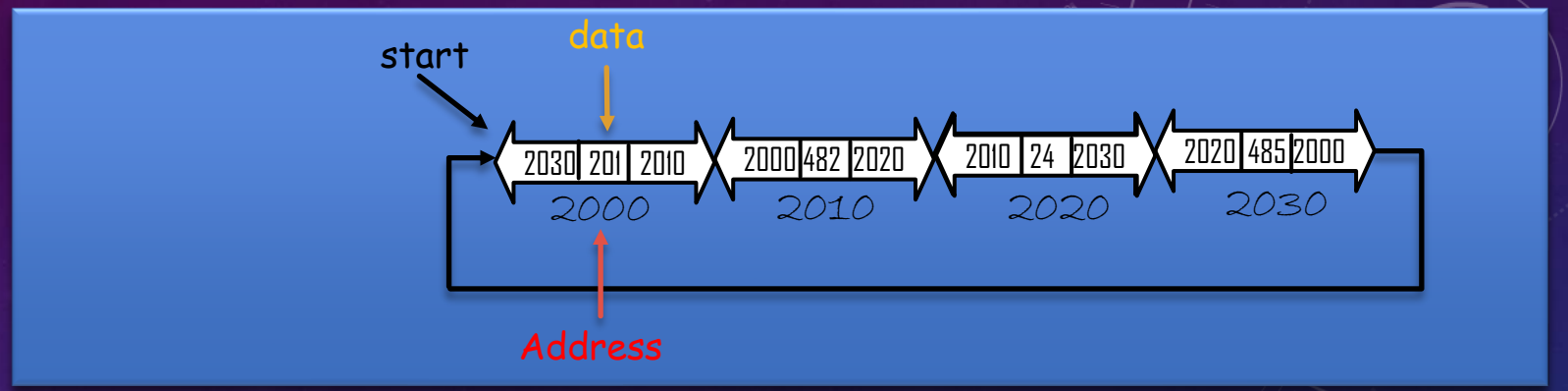
Step 3: start->prev->next=start->next;

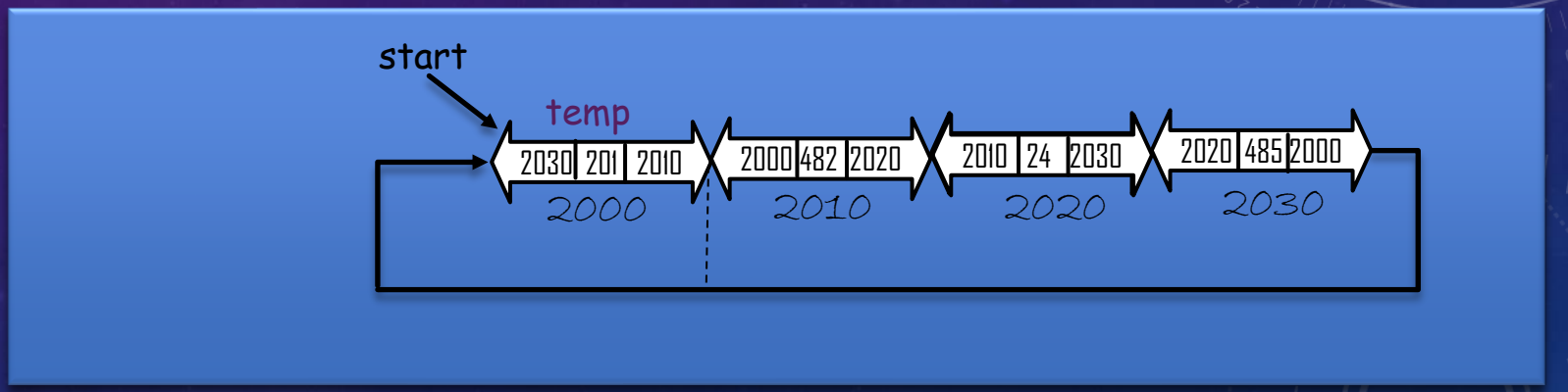
Step 4: start->next->prev=start->prev;

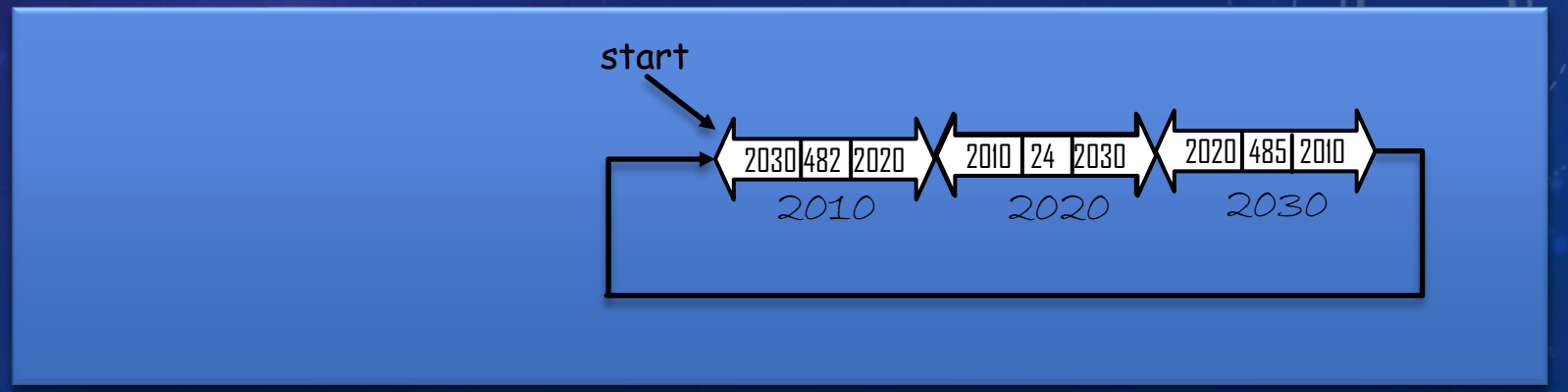
start=start->next;

Step 5: free(temp);

Step 6: STOP







DOUBLE CIRCULAR LINKED LIST

Deletion from Position/Middle

Step 1: START

Step 2: Read position i.e. "pos"

Step 3: q=start;

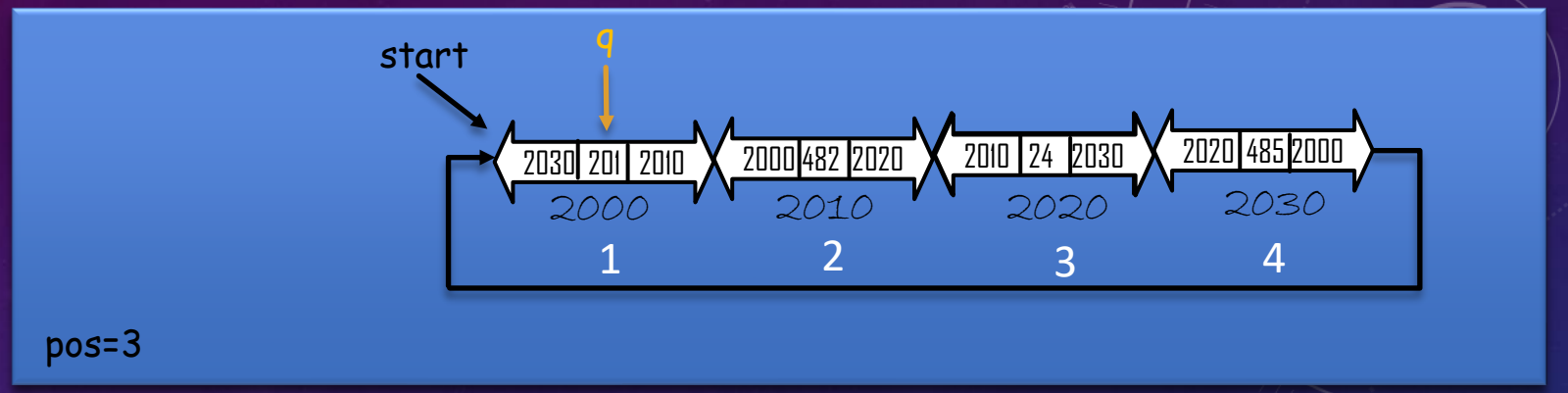
```
for(i=1;i<pos;i++)
{
    q=q->next;
}
```

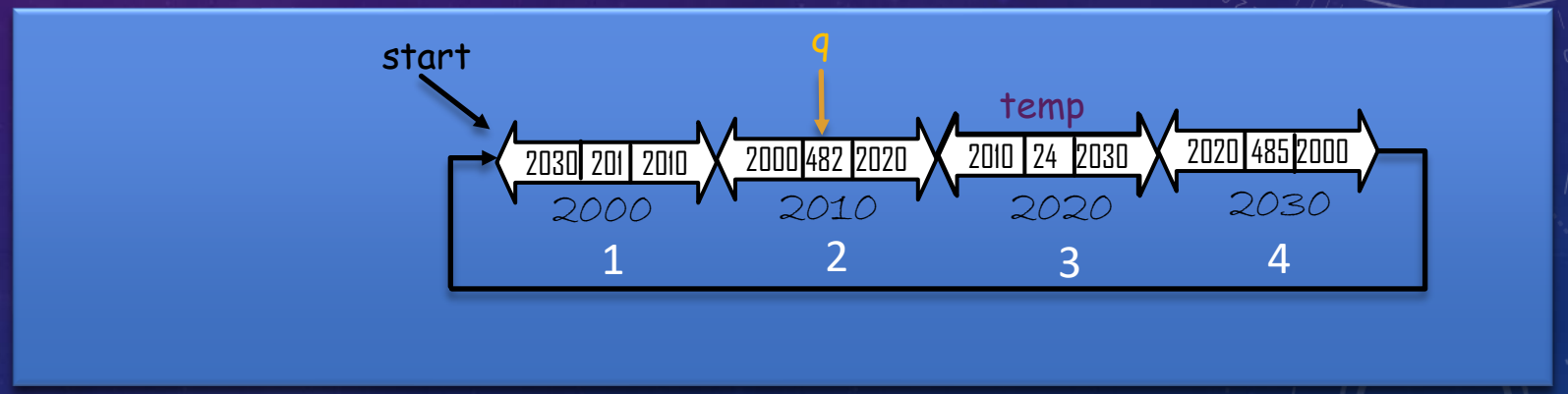
Step 4: temp=q->next;

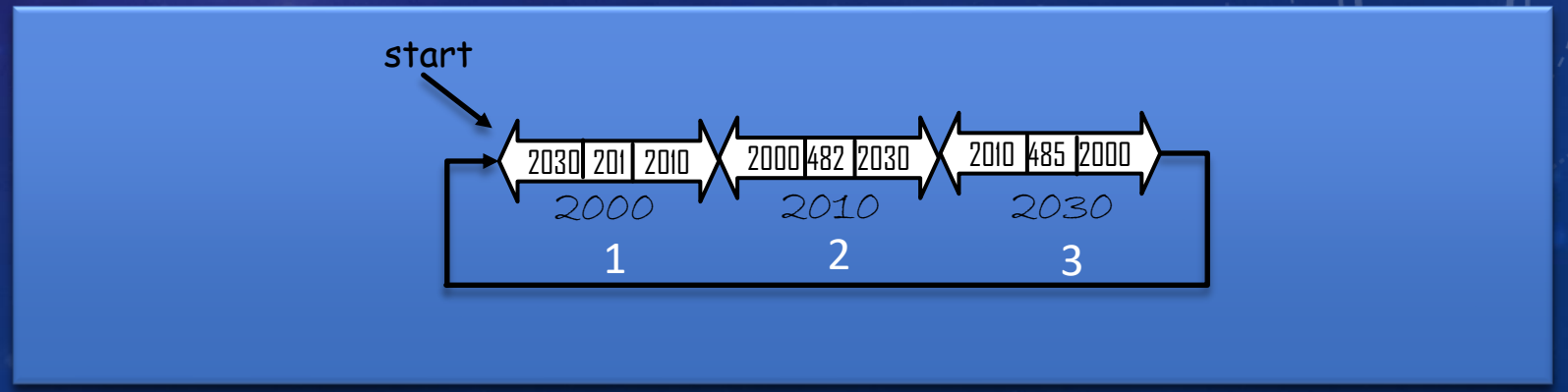
```
q->next=temp->next;
temp->next->prev=q;
```

Step 5: free(temp);

Step 6: STOP







DOUBLE CIRCULAR LINKED LIST

Deletion from End

Step 1: START

Step 2: q=start

Step 3: while(q->next->next!=start)

{

q=q->next;

}

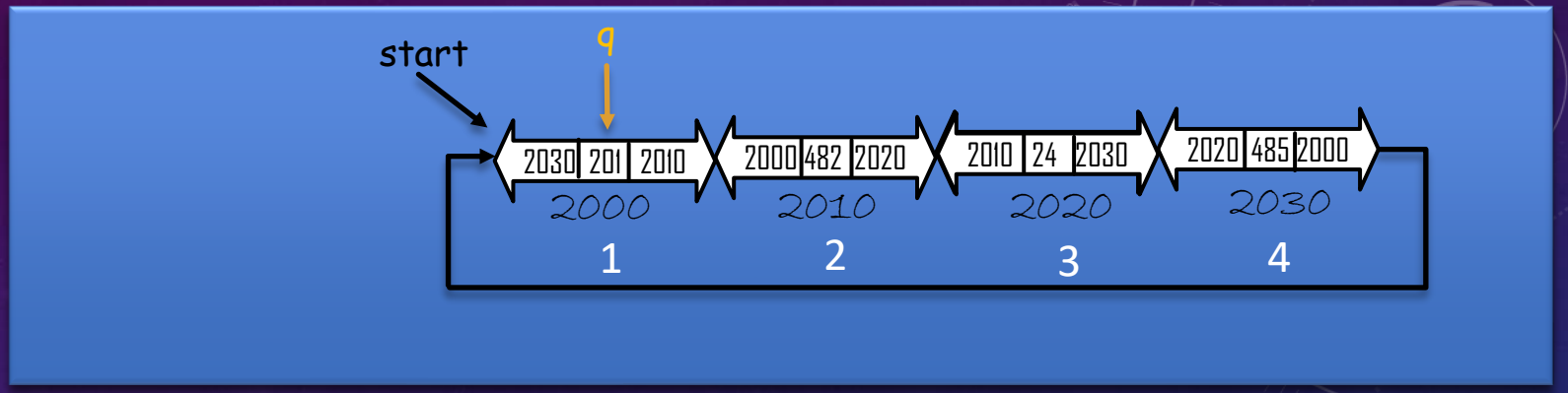
Step 4: temp=q->next;

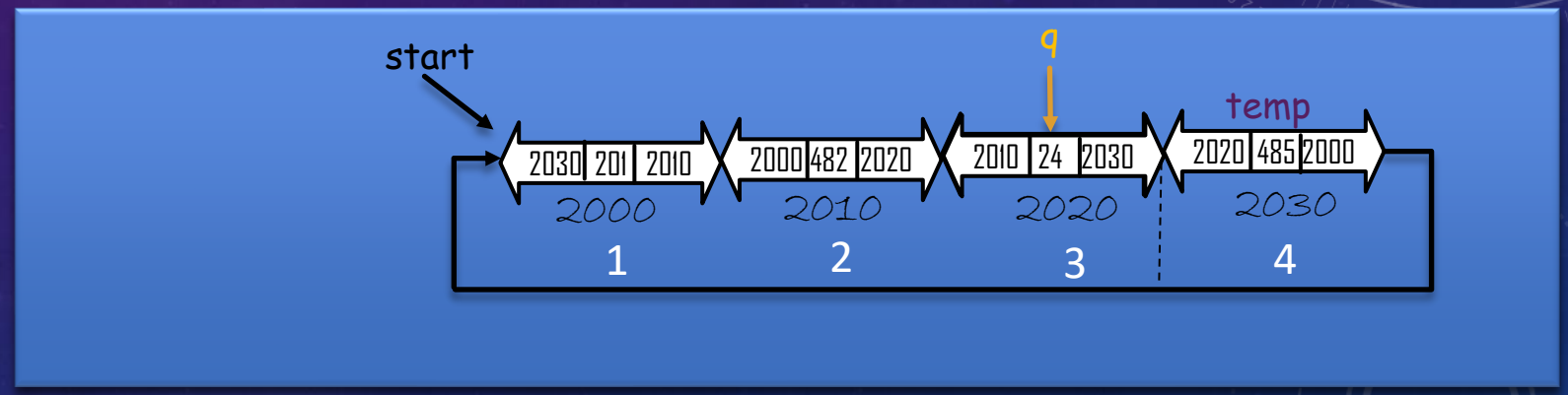
q->next=start;

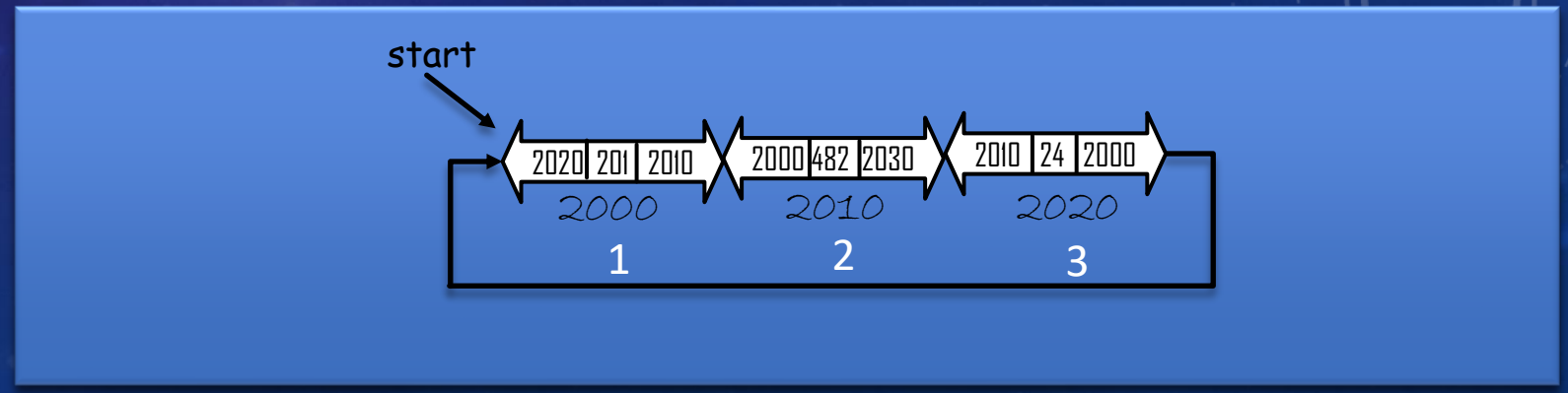
start->prev=q;

Step 5: free(temp);

Step 6: STOP







DOUBLE CIRCULAR LINKED LIST

Display Operation

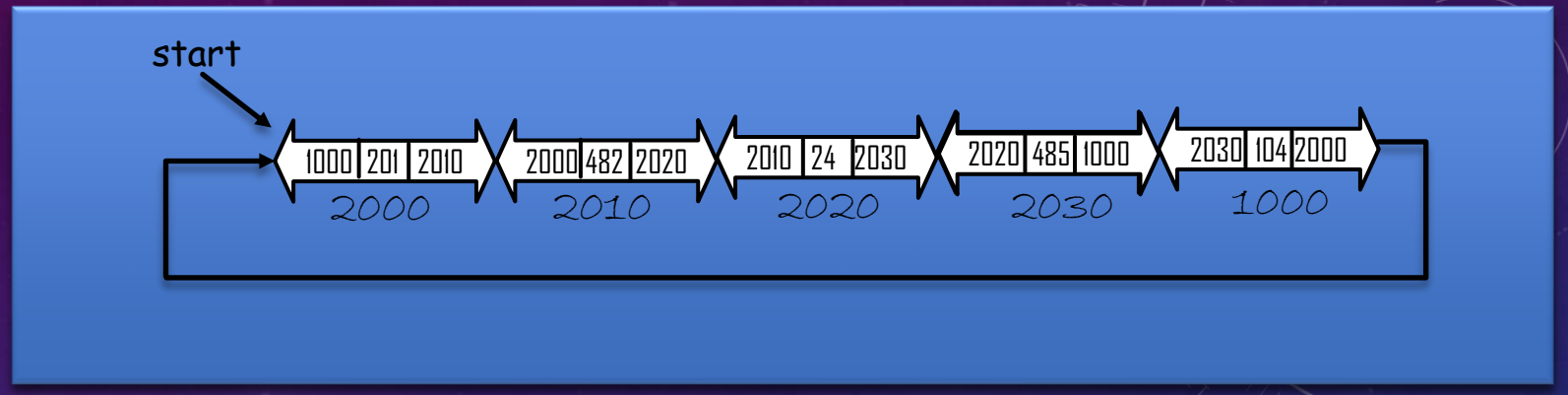
Step 1: START

Step 2: q=start

Step 3: while(q->next!=start)

```
{
    cout<<q->data<<" -> ";
    q=q->next;
}
```

Step 4: STOP



Thank you

Made By: Yagneswara Naidu (CSE)