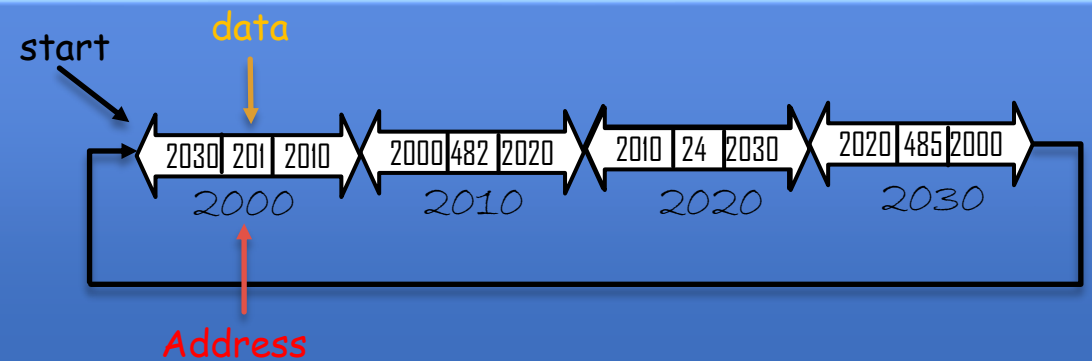
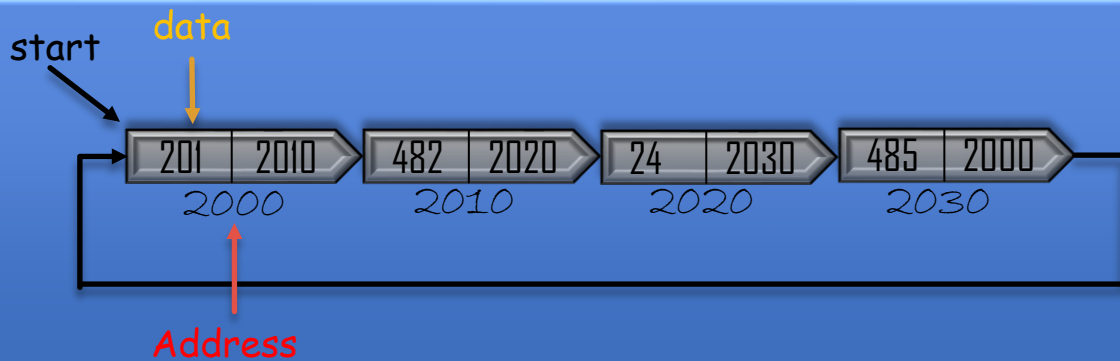


CIRCULAR LINKED LIST

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”

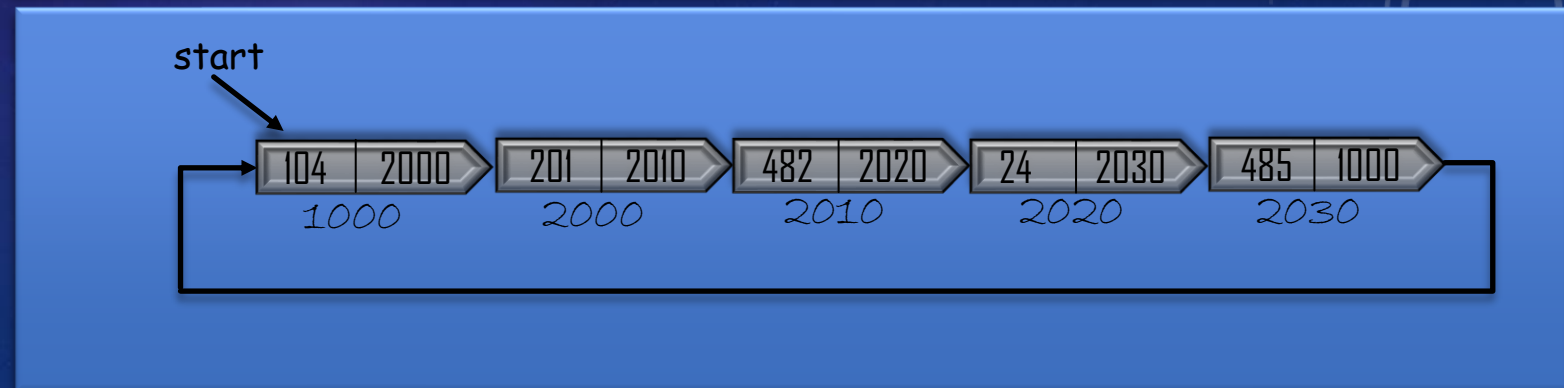
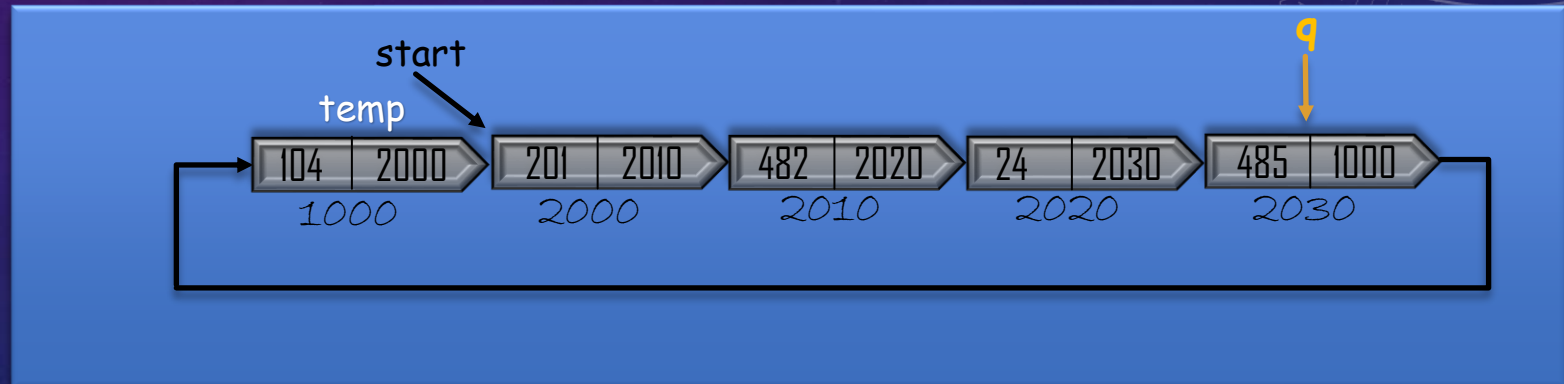
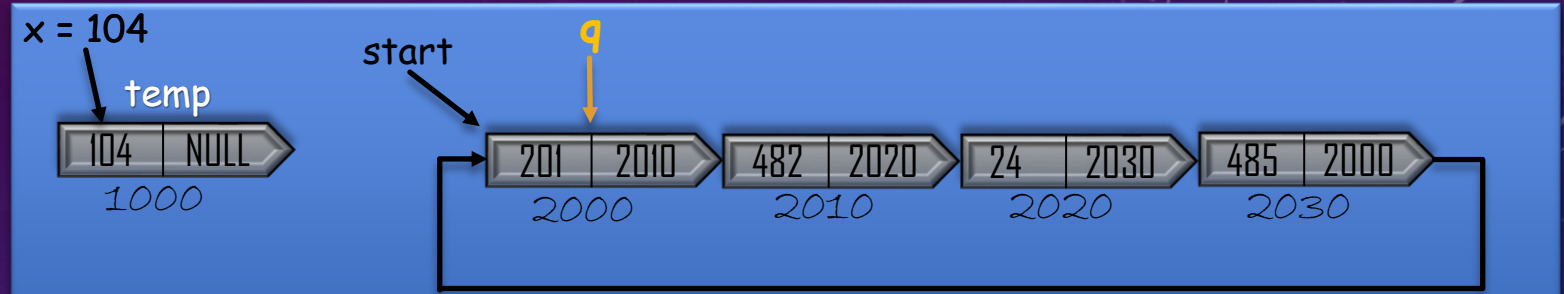
Step 3: Read data “x”
temp->data=x;

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++)

{

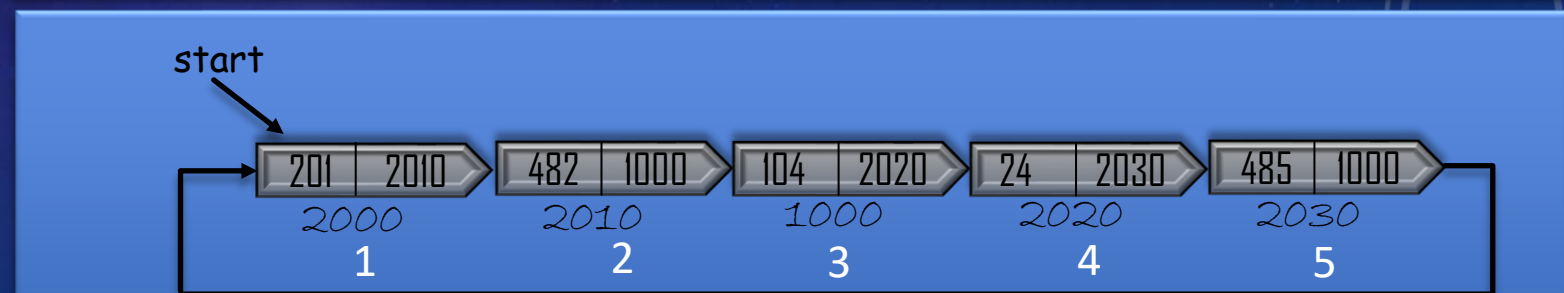
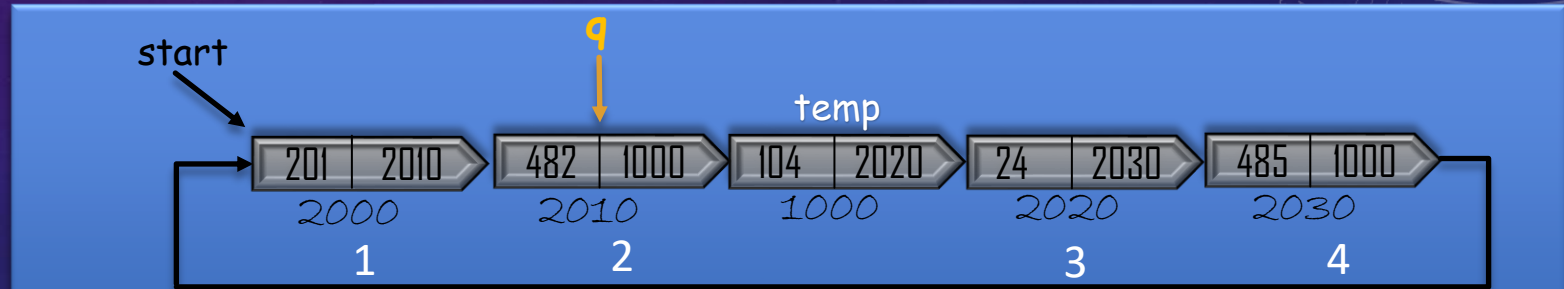
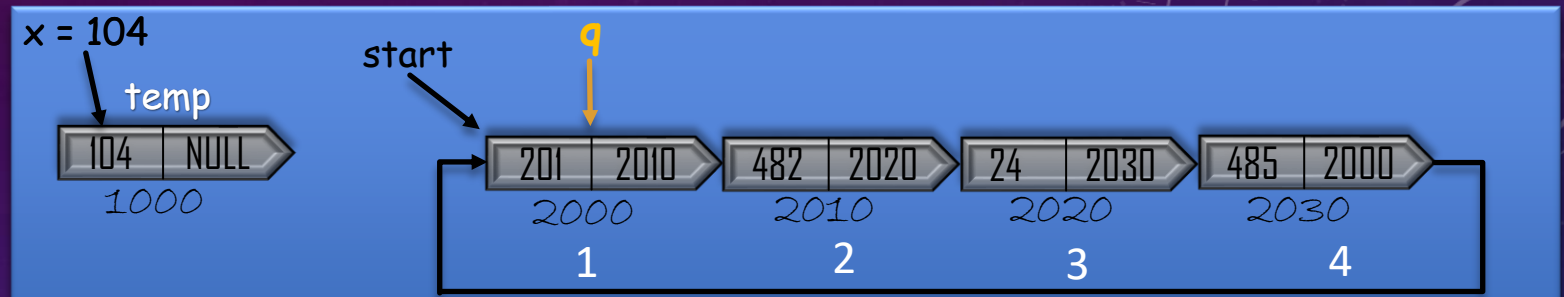
q=q->next;

}

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”

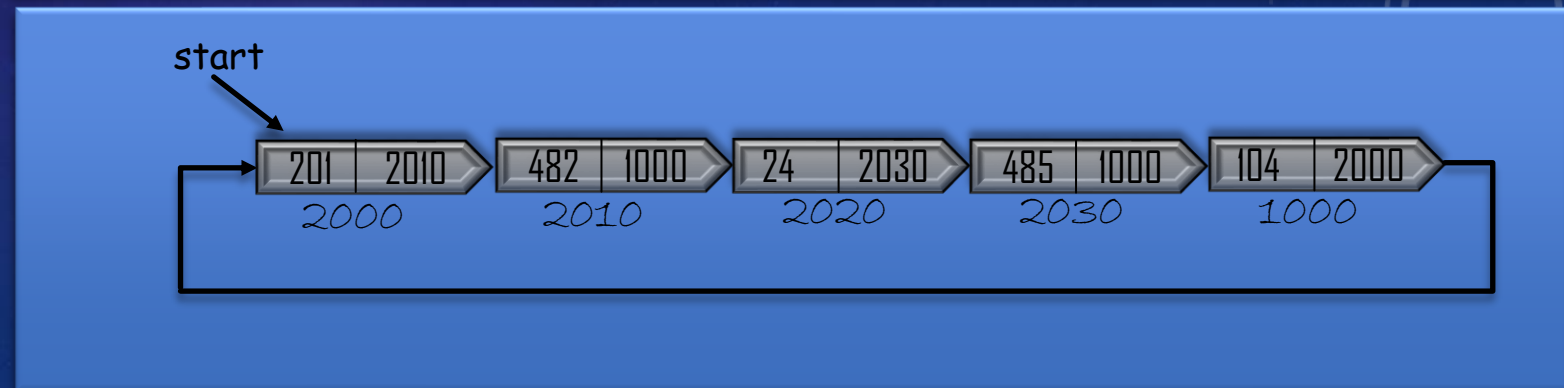
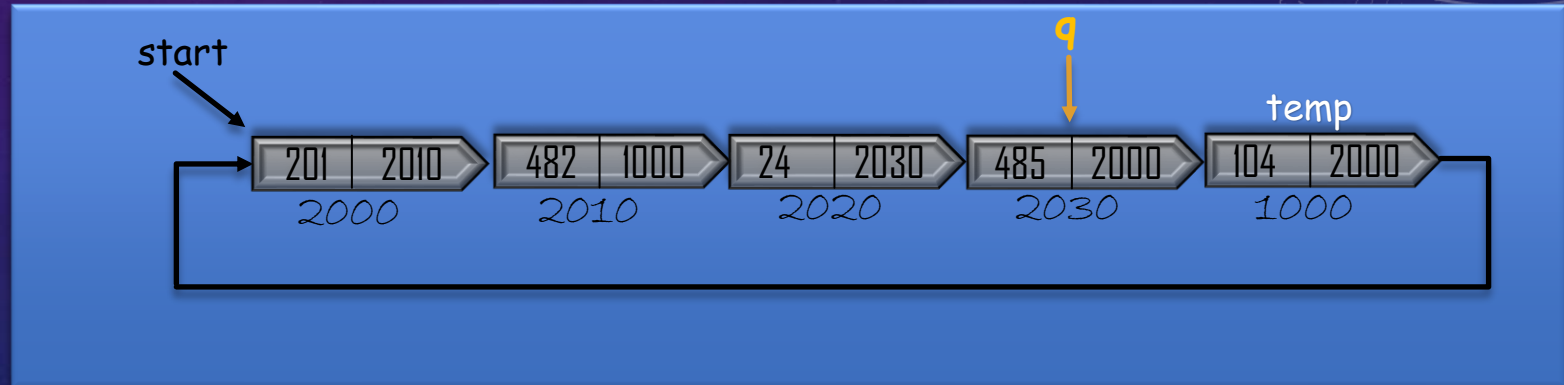
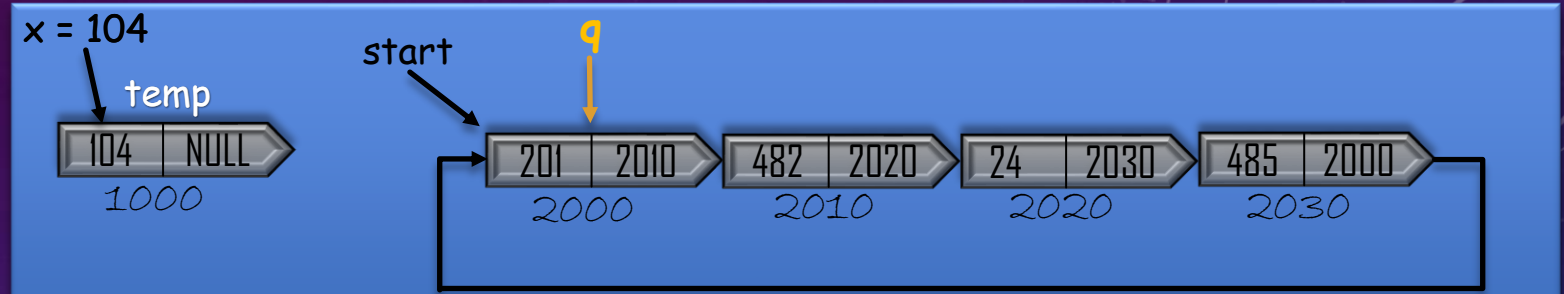
Step 3: Read data “x”
temp->data=x;

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

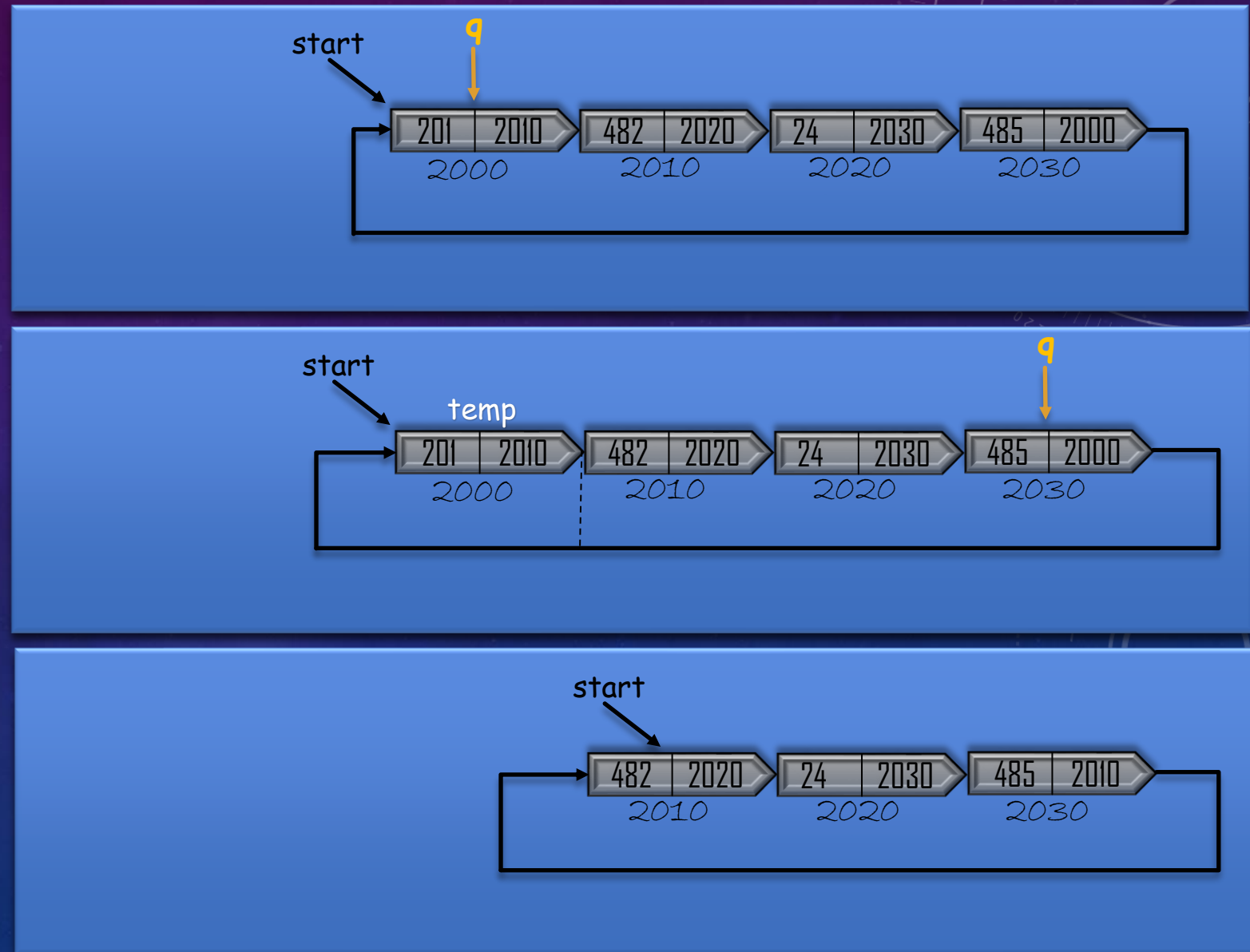
Step 3: q=start
while(q->next!=start)
{
 q=q->next;
}

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;
```

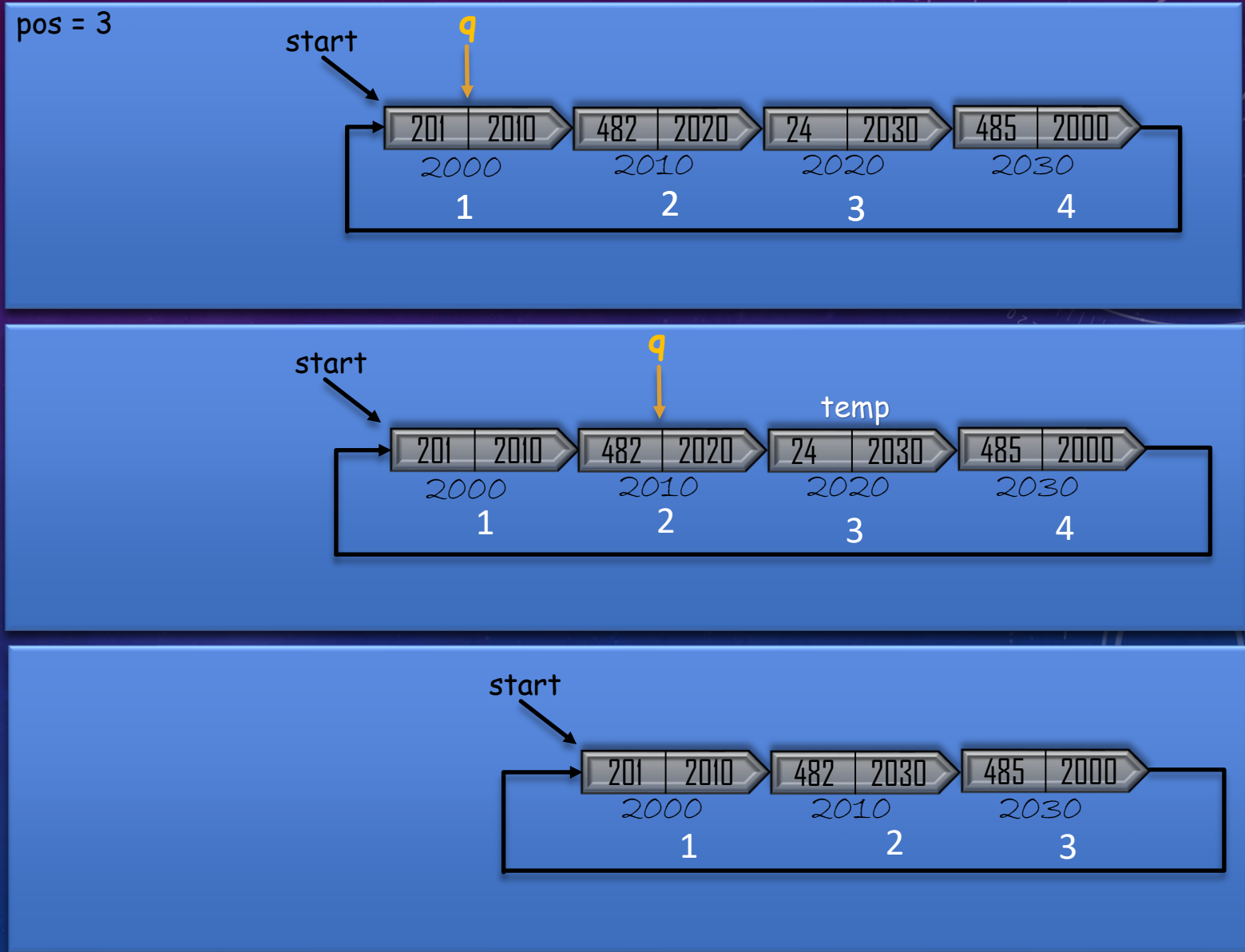
```
}
```

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: $q = \text{start}$

Step 3: $\text{while}(q \rightarrow \text{next} \rightarrow \text{next} \neq \text{start})$

{

$q = q \rightarrow \text{next};$

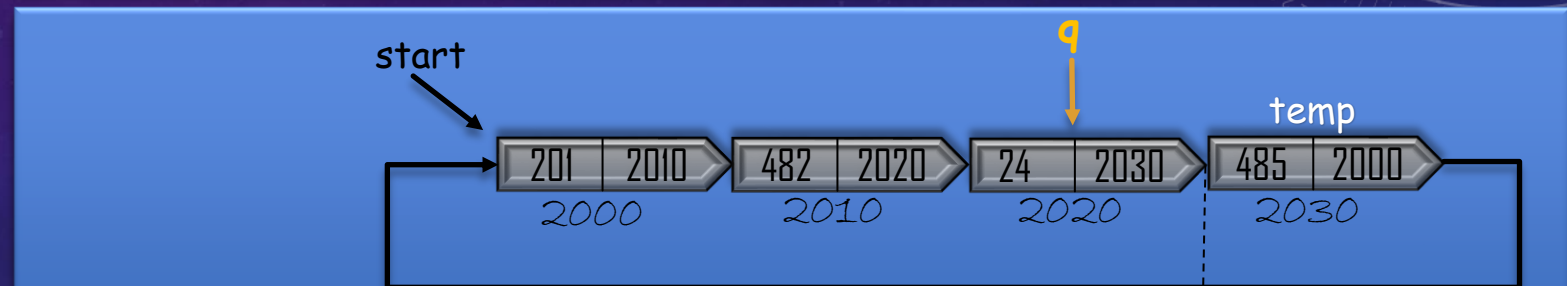
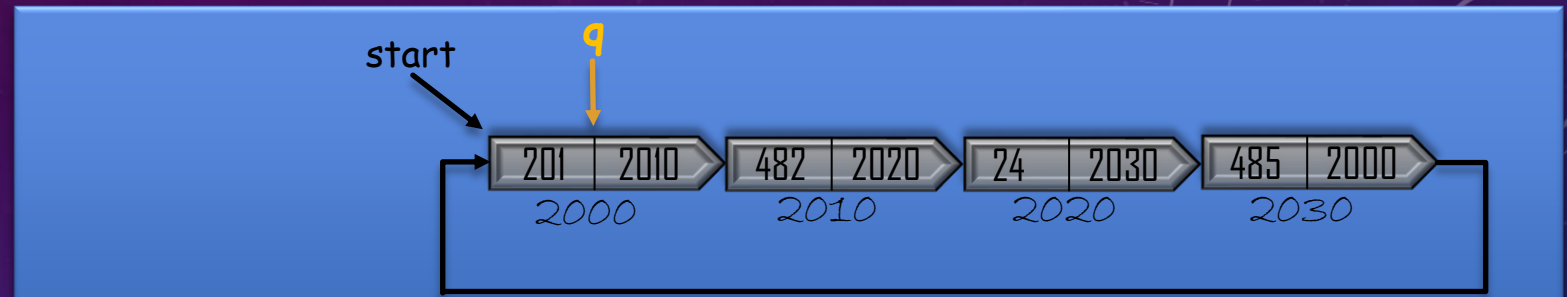
}

Step 4: $\text{temp} = q \rightarrow \text{next};$

$q \rightarrow \text{next} = \text{start};$

Step 5: $\text{free}(\text{temp});$

Step 6: STOP



SINGLE CIRCULAR LINKED LIST

Display Operation

Step 1: START

Step 2: $q = \text{start}$

Step 3: while($q \rightarrow \text{next} \neq \text{start}$)

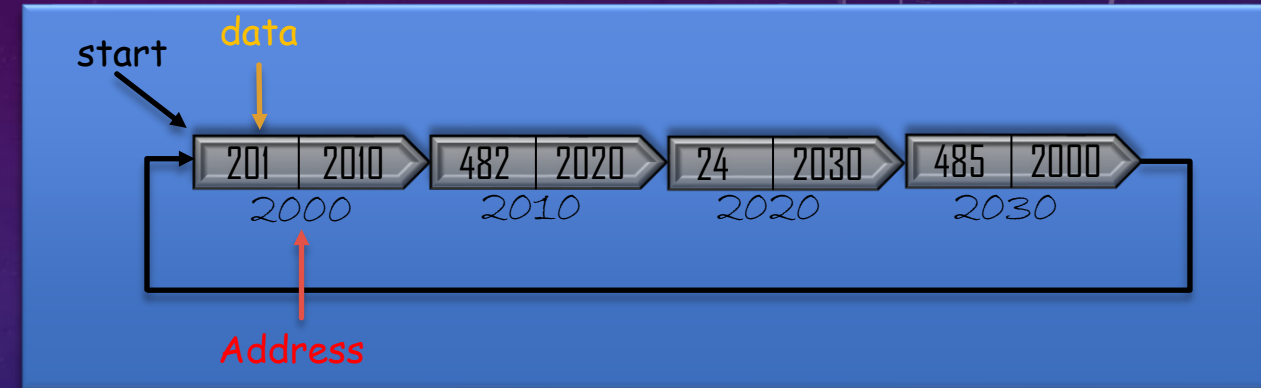
{

$\text{cout} \ll q \rightarrow \text{data} \ll " -> ";$

$q = q \rightarrow \text{next};$

}

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;

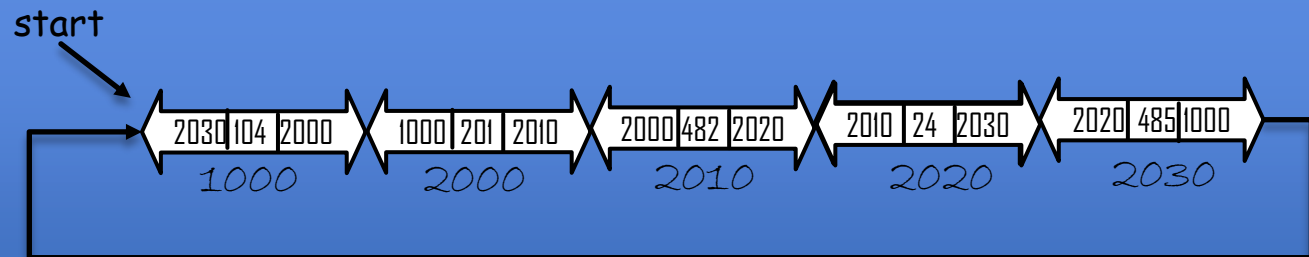
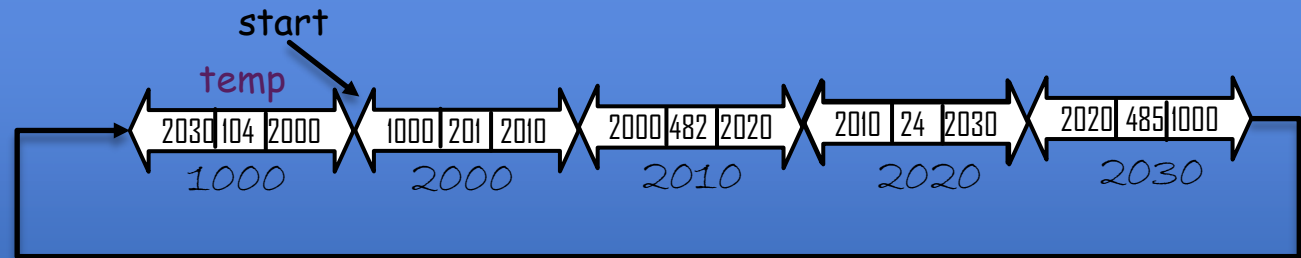
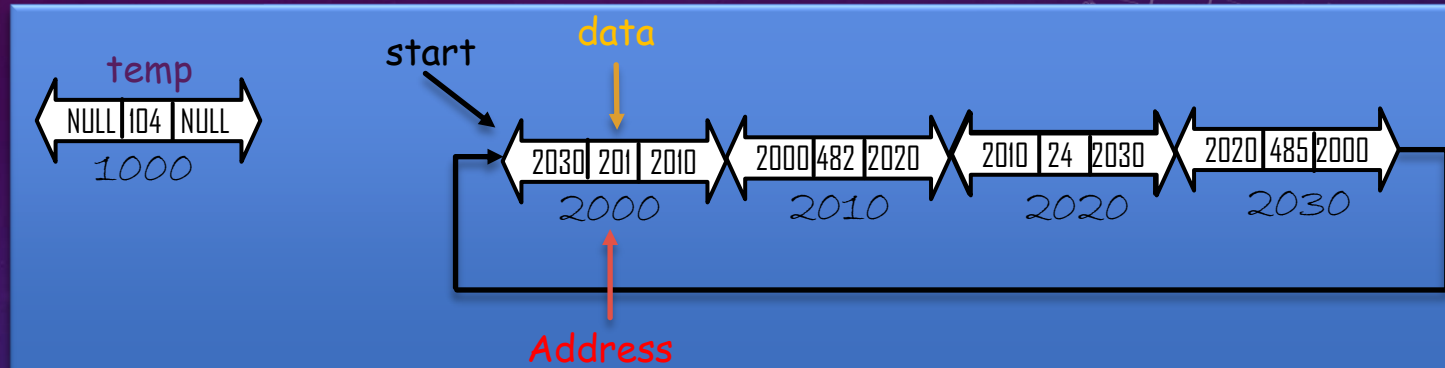
temp->prev=start->prev;

temp->next=start;

start->prev=temp;

Step 5: start=temp;

Step 6: STOP



DOUBLE CIRCULAR LINKED LIST

Insertion at Position/Middle

Step 1: START

Step 2: Create a new node i.e. “temp”

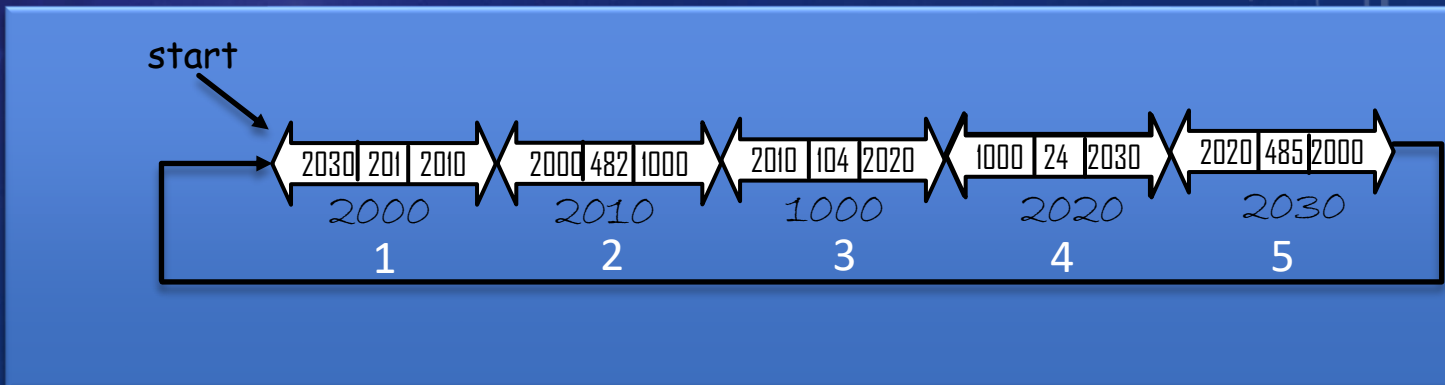
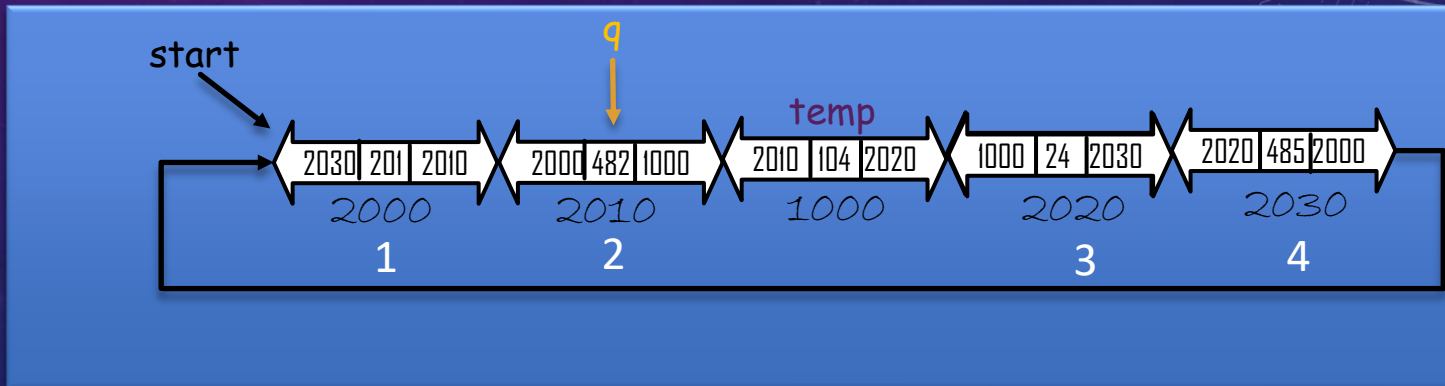
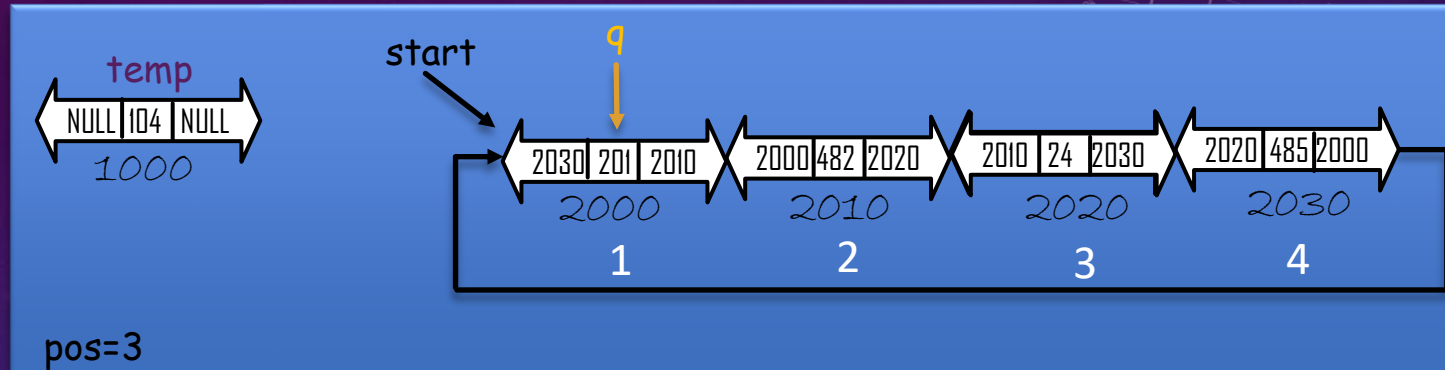
Step 3: Read data “x”
temp->data=x;
temp->next=NULL;
temp->prev=NULL;

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;

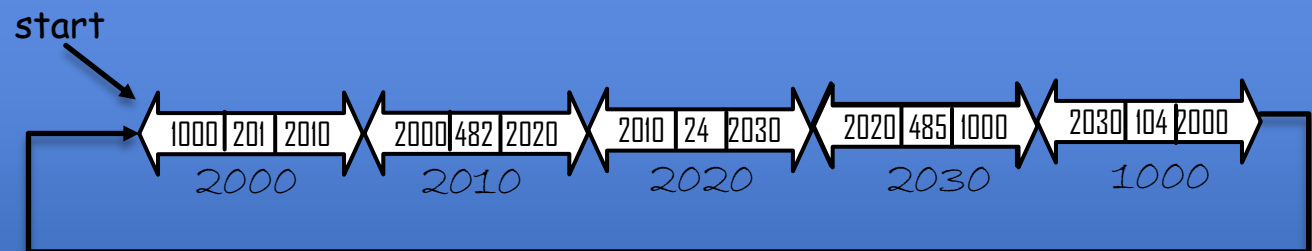
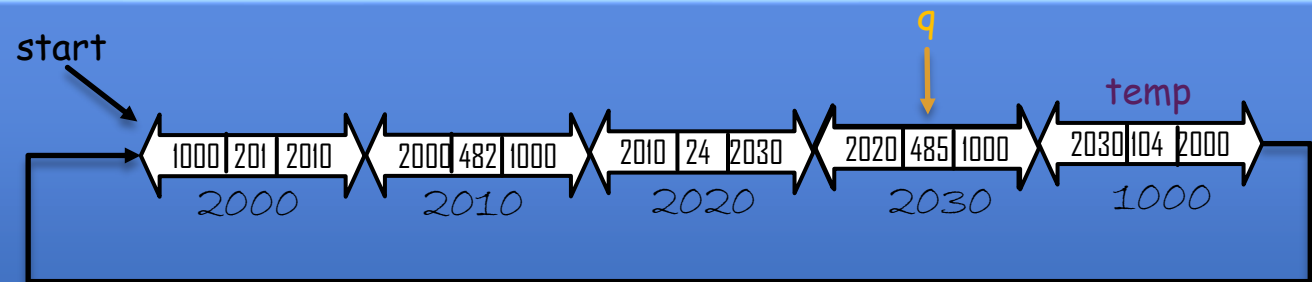
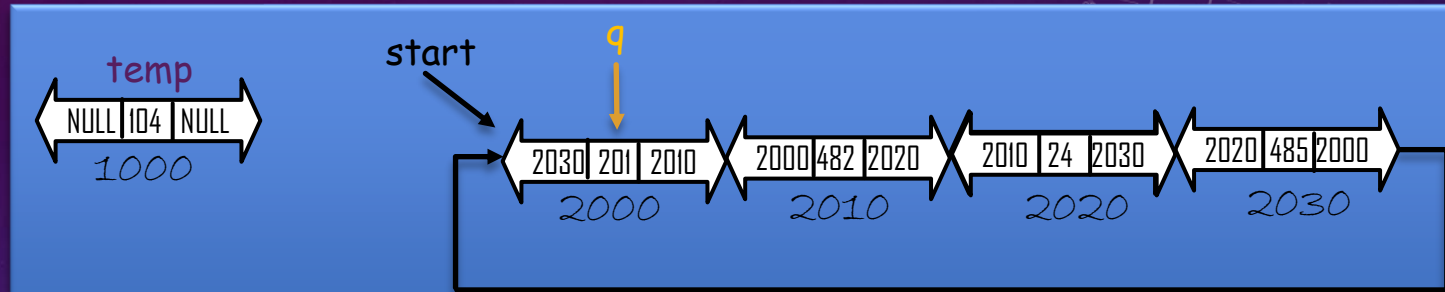
Step 4: q=start
while(q->next!=start)
{
 q=q->next;
}

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

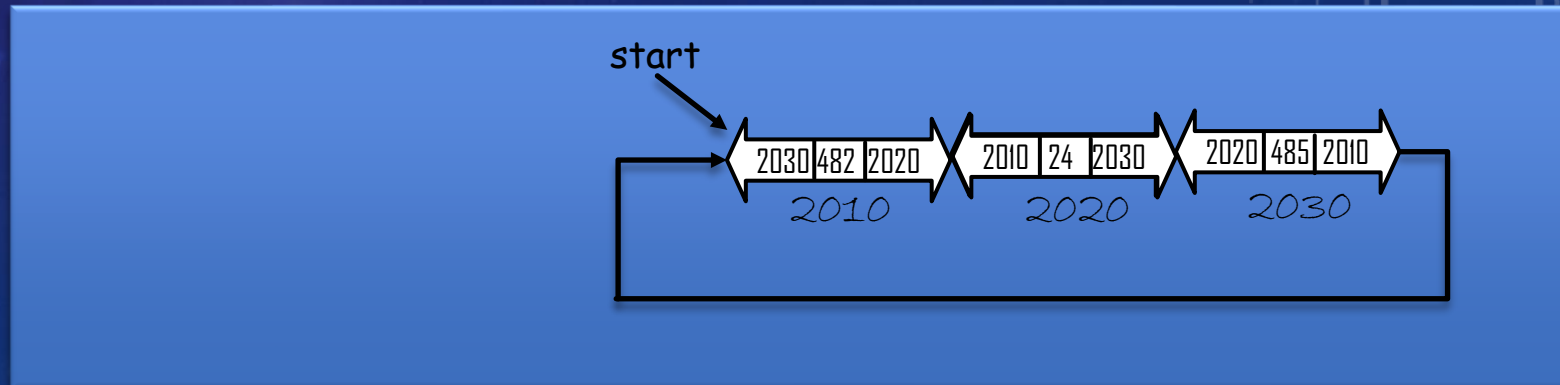
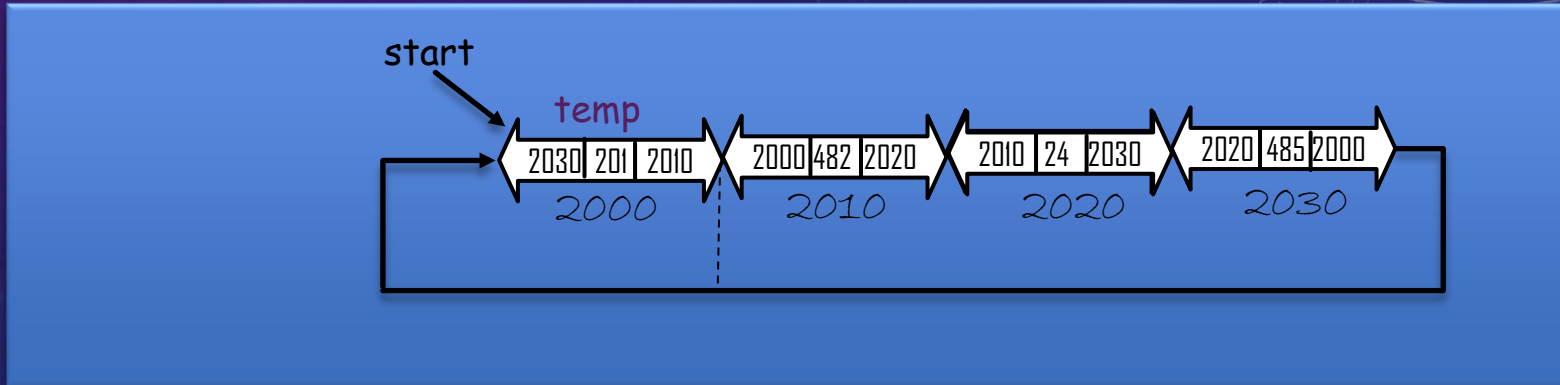
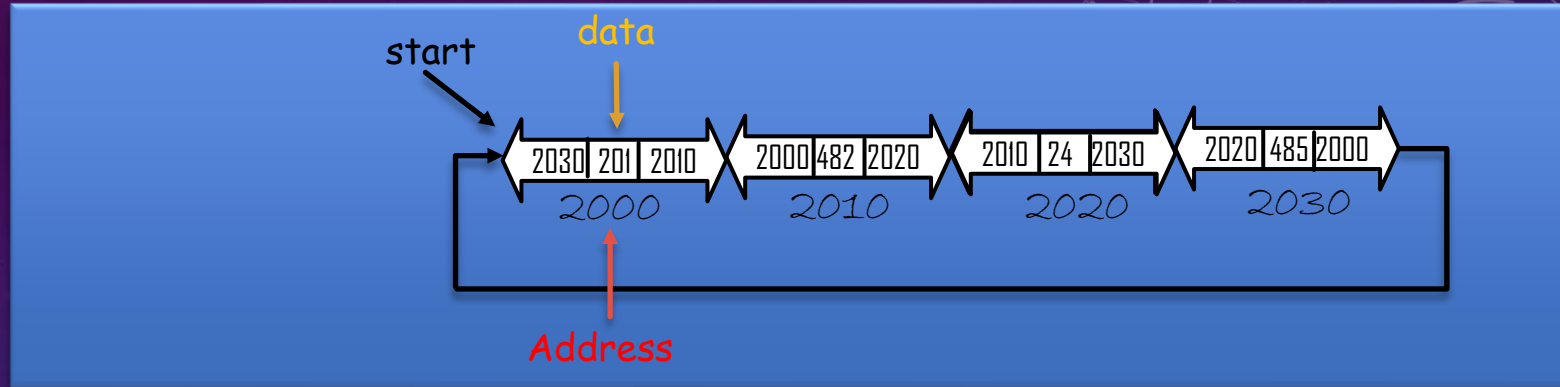
Step 2: temp=start;

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;
```

```
}
```

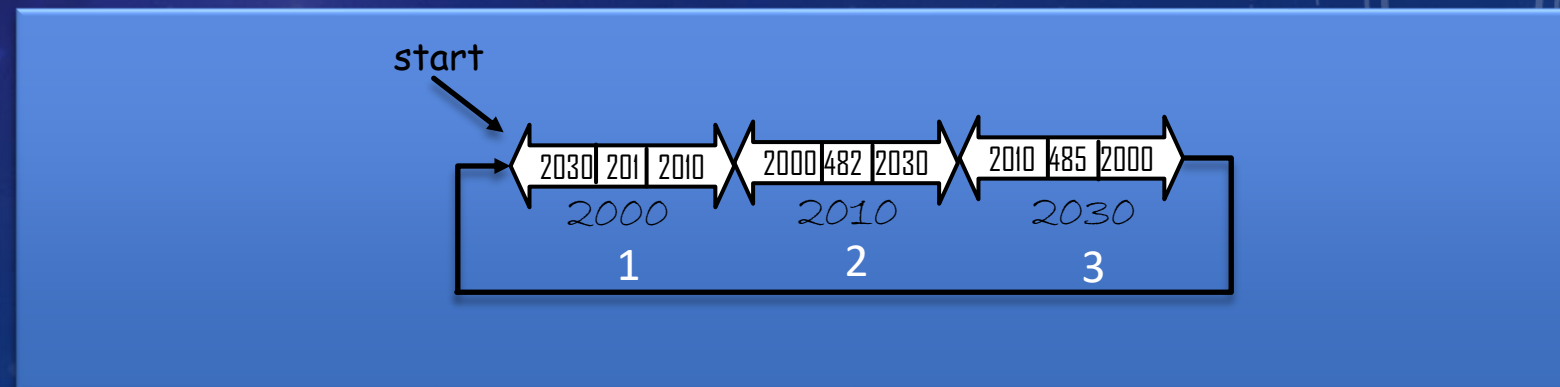
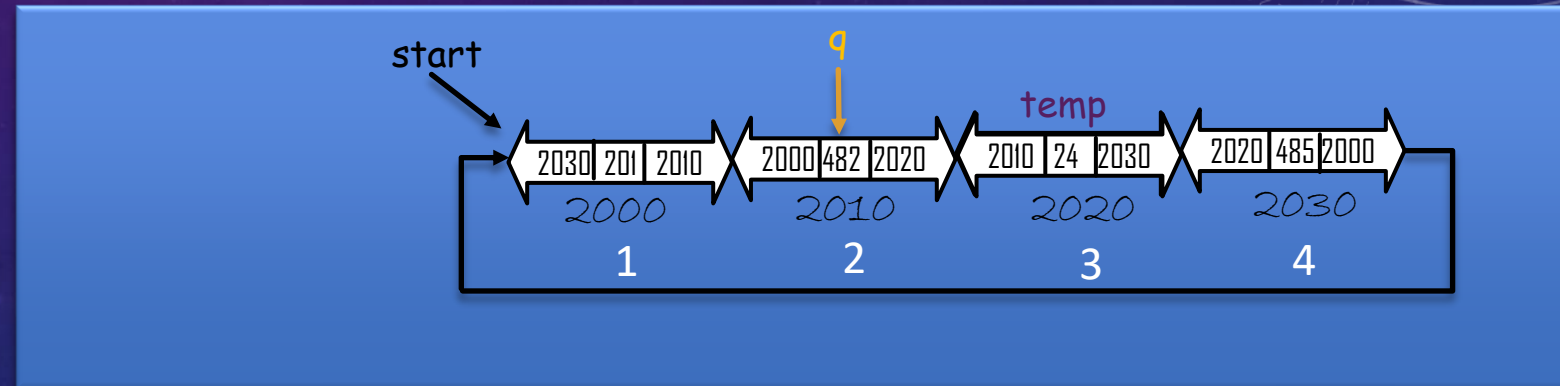
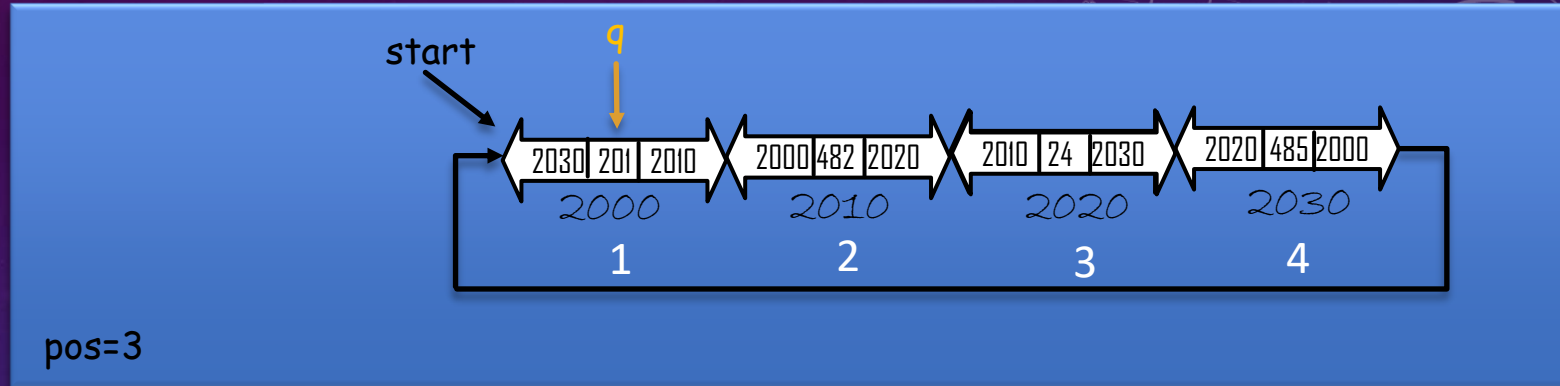
Step 4: temp=q->next;

```
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: $q = \text{start}$

Step 3: $\text{while}(q \rightarrow \text{next} \rightarrow \text{next} \neq \text{start})$

{

$q = q \rightarrow \text{next};$

}

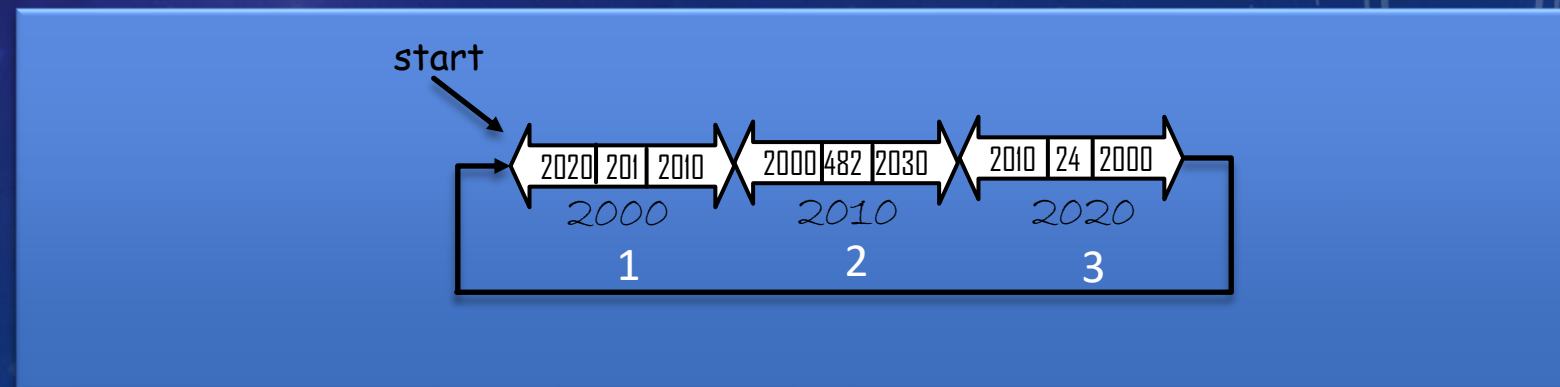
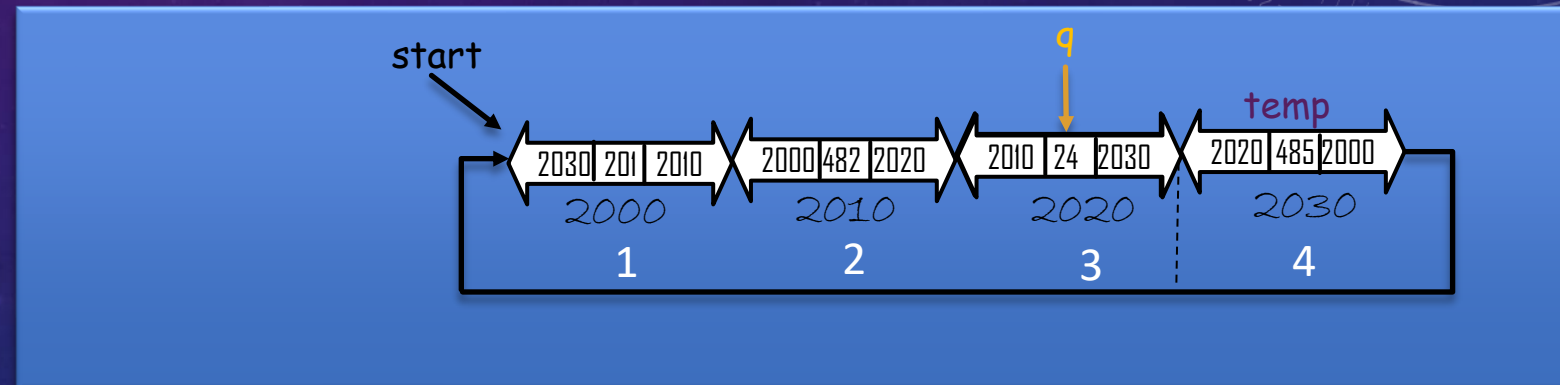
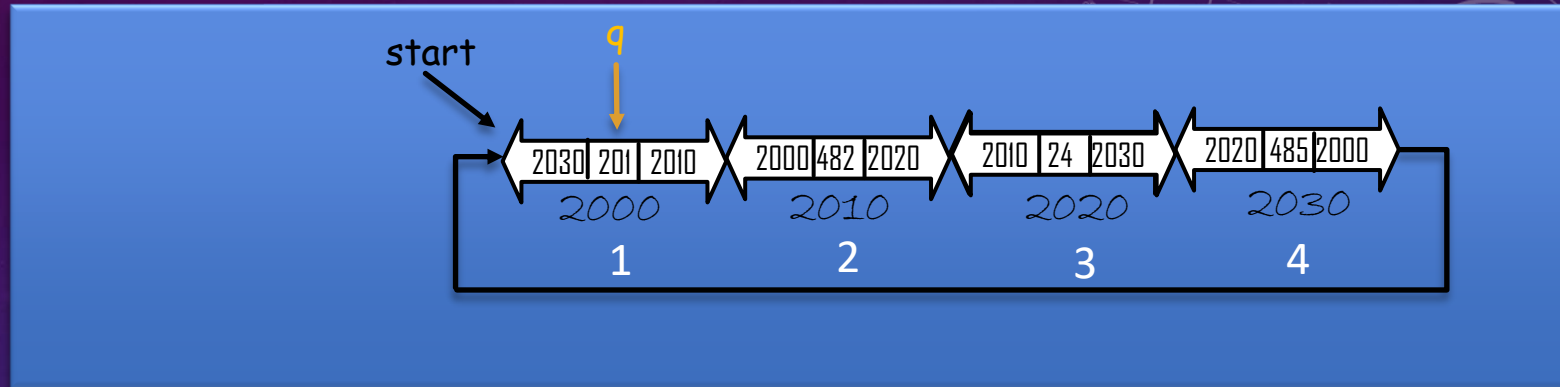
Step 4: $\text{temp} = q \rightarrow \text{next};$

$q \rightarrow \text{next} = \text{start};$

$\text{start} \rightarrow \text{prev} = q;$

Step 5: $\text{free}(\text{temp});$

Step 6: STOP



DOUBLE CIRCULAR LINKED LIST

Display Operation

Step 1: START

Step 2: q=start

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

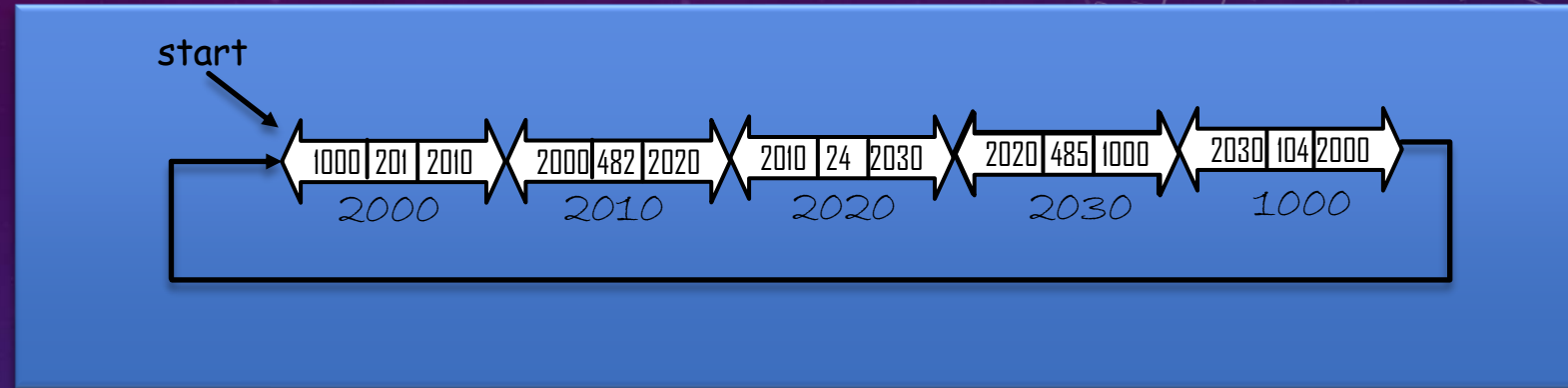
{

cout<<q->data<<" -> ";

q=q->next;

}

Step 4: STOP



The background is a deep blue gradient with a subtle pattern of white stars and faint, glowing circular lines. On the right side, there are several concentric circles and arcs, some with tick marks and numbers, resembling technical or scientific diagrams. The text "Thank you" is prominently displayed in the center in a bold, golden-yellow, 3D-style font. Below it, the words "Thank you" are repeated in a smaller, semi-transparent, dark blue font, creating a layered effect.

Thank you
Thank you