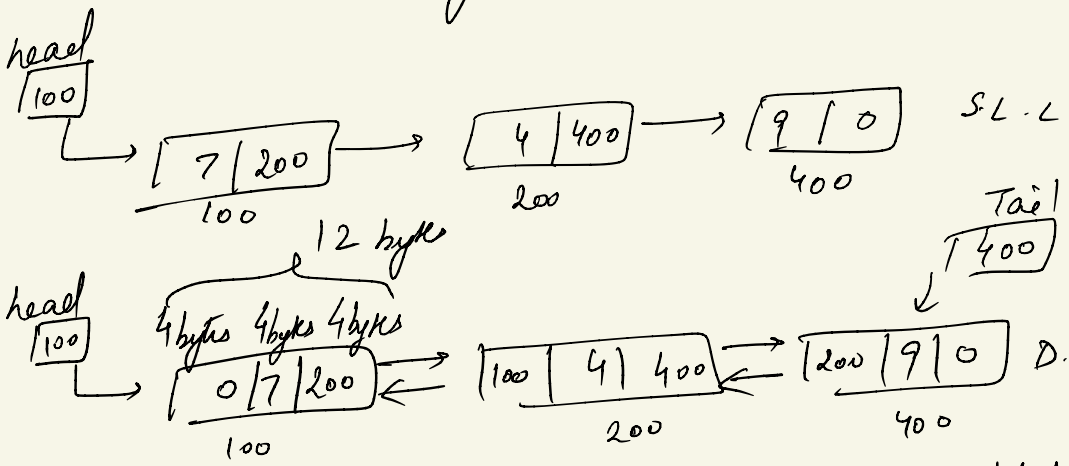
Data Structures & Algorithms - I

Doubly linked list

Dr. Ruana Mukherjee
Sec - B

Introduction to Doubly Linked list



→ Instead of 1 pointer (next) in S.L.L maintain 2 pointers (next, prev) in D.L.L

→ Traverse the L.L in forward or reverse direction

→ Tail ⇒ O(N) ⇒ O(1)

Implementation of D.L.L

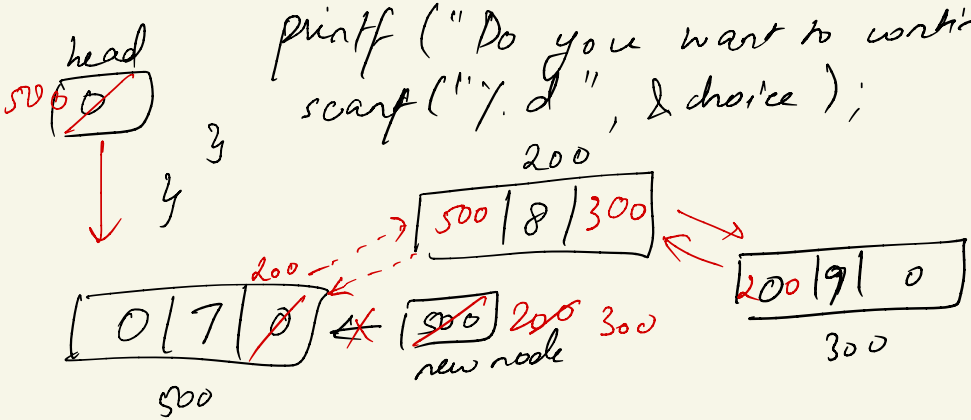
```
struct node
{ int data;
  struct node *next, *prev;
};
struct node *head, *newnode;
```

```
void createDLL()
{ head = 0; struct node * temp; int choice = 1;
while (choice)
{newnode = (struct node *) malloc (sizeof (struct node));
   printf ("Enter data");
   scanf ("%d", &newnode -> data);
   newnode -> prev = 0;
   newnode -> next = 0;
   if (head == 0)
   {head = newnode;
   }
   else
      { head -> next = newnode;
        newnode -> prev = head;
      }
   printf ("Do you want to continue?")
   scanf ("%d", &choice);
}
```

or `*newnode.data`

`head = temp = newnode;`

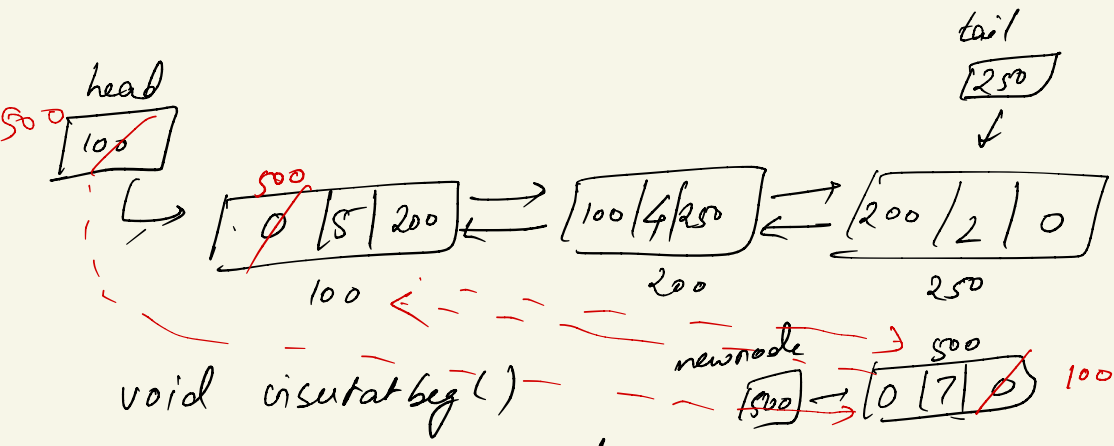
`temp->next = newnode;`
`newnode->prev = temp;`
`temp = newnode;`



```
void displayDLL()
  { struct node *temp;
    temp = head;
   while (temp != 0)
{printf("%d", temp->data);
    temp = temp->next;
}
}
  void main()
     { create();
       display();
       getch();
      }
```

Insertion into DLL

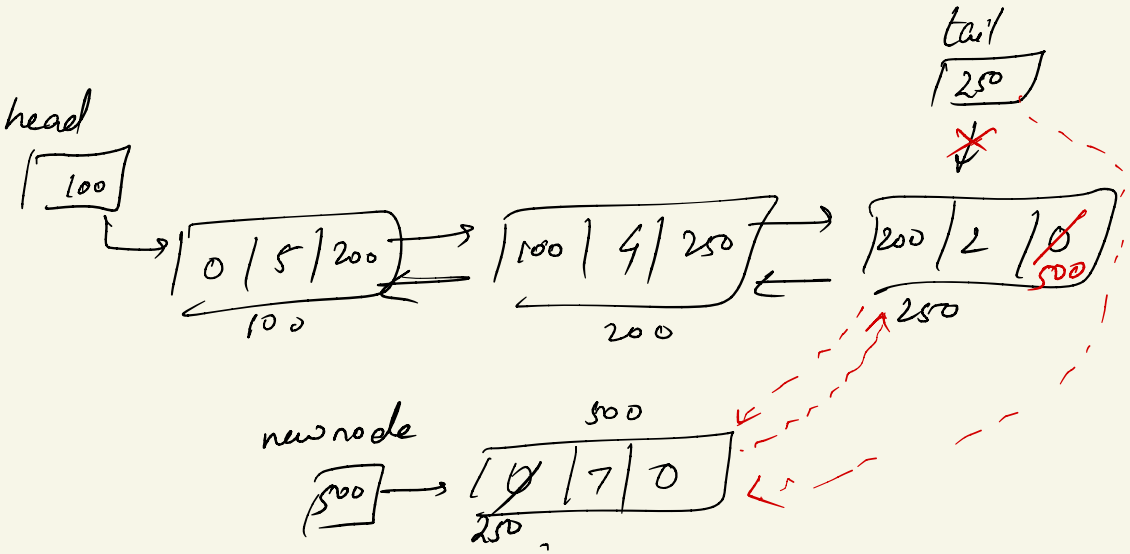
insert at beg ( )
insert at end ( )
insert at pos ( )
insertafterpos ( )




```
void insertatbeg()
{ struct node * newnode;
  newnode = (struct node *) malloc (sizeof (struct node));
  printf (" Enter data ");
  scanf (" %d ", & newnode -> data);
    newnode -> prev = 0;
    newnode -> next = 0;
    head -> prev = newnode;
      newnode -> next = head;
      head = newnode;
}
```

```
void insert at end ( )
{
    struct node * new node ;
    new node = (struct node *) malloc (size of
                                       (struct node));

    Printf ("Enter data ");
    scanf (" %d ", &newnode -> data);
        newnode -> prev = 0;
        newnode -> next = 0;
        tail -> next = newnode;
        newnode -> prev = tail;
        tail = newnode;
}
```

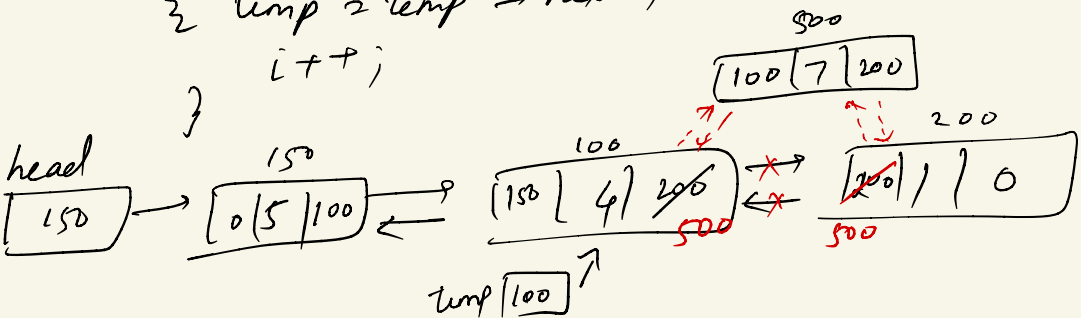


```
void insert at pos ()
{ int pos;
  printf ("Enter position ");
  scanf ("%d", &pos);
  if (pos < 1 && pos > Length)
  { printf ("Invalid pos ");
  }
  else if (pos == 1)
  { insert at beg ();
  }
  else
  { struct node * newnode, * temp; temp = head;
    newnode = (struct node *) malloc (sizeof
                (struct node));
    printf ("Enter data ");
    scanf ("%d", & newnode -> data);
    while (i < pos - 1)
    { temp = temp -> next;
      i++;
    }
    newnode -> prev = temp;
    newnode -> next = temp -> next;
    temp -> next = newnode
    newnode -> next -> prev = newnode;
  }
```

Ques Assignment

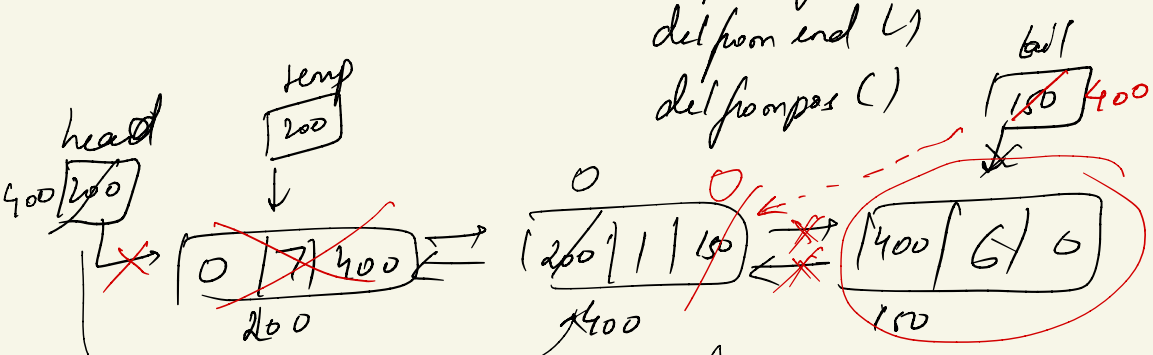
void insert after pos()

pos=3



Deletion from DLL

del from beg ()
del from end ()
del from pos ()

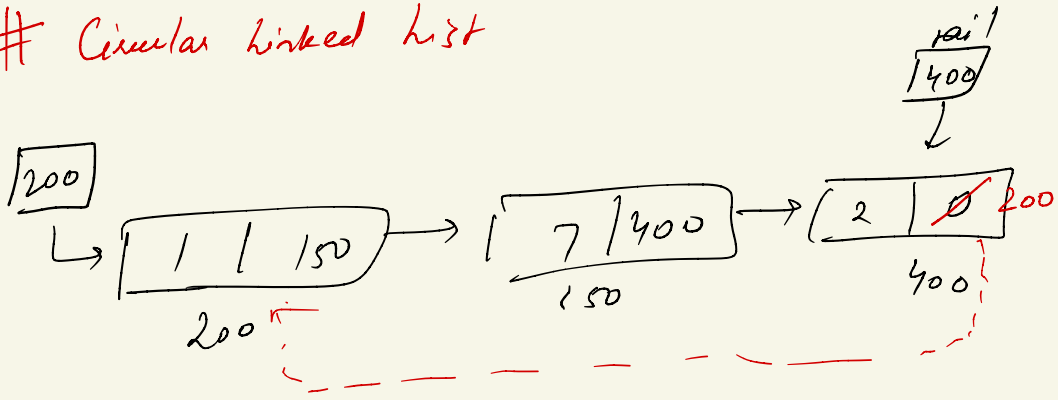



```
Void delfrombeg ()
{ struct node *temp;
  if (head == 0)
    { printf ("List is empty");
    }
  else
    { temp = head;
      head = head -> next;
      head => prev = 0;
      free (temp);
    }
}
```

```
void delfromend()
{ struct node
    *temp;
  if (tail == 0)
  { printf("Empty
      List");
  }
  else
  { temp = tail;
    tail -> prev -> next
        = 0;
    tail = tail -> prev;
    free (temp);
  }
}
```

Class Assignment
void delfrompos ()

Circular linked list



Ques Assignment

void createCLL ()

void insertCLL atbeg ()
insertCLL at end ()
insertCLL at pos ()

void delCLL frombeg ()
delCLL from end ()
delCLL at pos ()