//////////////////////////// SINGLY LINKED LIST //////////////////////////////////////////

Singly Linear Linked List

////////////////////////////////////////////////////////////////////////////// struct node

{

int Data;

struct node \*next;

};

///////////////////////////////////////////////////////////////////

//Insert node at first position

BOOL InsertFirst(struct node \*\*first, int value)

{

struct node \*newn=NULL;

newn = (struct node \*)malloc(sizeof(struct node));

newn -> data = value;

newn -> next = NULL;

if (\*first == NULL)

{

\*first = newn;

}

else

{

newn -> next = \*first;

\*first = newn;

}

return TRUE;

}

//////////////////////////////////////////////////////////////

//Display nodes of LL

void Display(struct node \*first)

{

while(first != NULL)

{

printf(" %d -> ",first -> data);

First = first -> next;

}

}

//////////////////////////////////////////////////////////////

//Count node

//traversal logic is same as that of display

int CountNode (struct node \*first)

{

int Count = 0;

while(First != NULL)

{

Count++;

first = first -> next;

}

return count;

}

//////////////////////////////////////////////////////////////////////////////////

//search first occurance of desired data in given LL

int SearchFirstOccurance(struct node \*first, int no)

{

int pos = 0;

while( First != NULL )

{

pos++;

if(first -> data == no)

{

break;

}

first = first -> next;

}

If(first==NULL)

{

Return 0;

}

Else

{

Return pos;

}

}

///////////////////////////////////////////////////////////////////////////////

//Delete 1st node from given LL

BOOL DeleteFirst( struct node \*\*first)

{

struct node \*temp;

temp = \*first;

if(\*first == NULL)

{

return TRUE;

}

else

{

\*first = \*first -> Next;

free(temp);

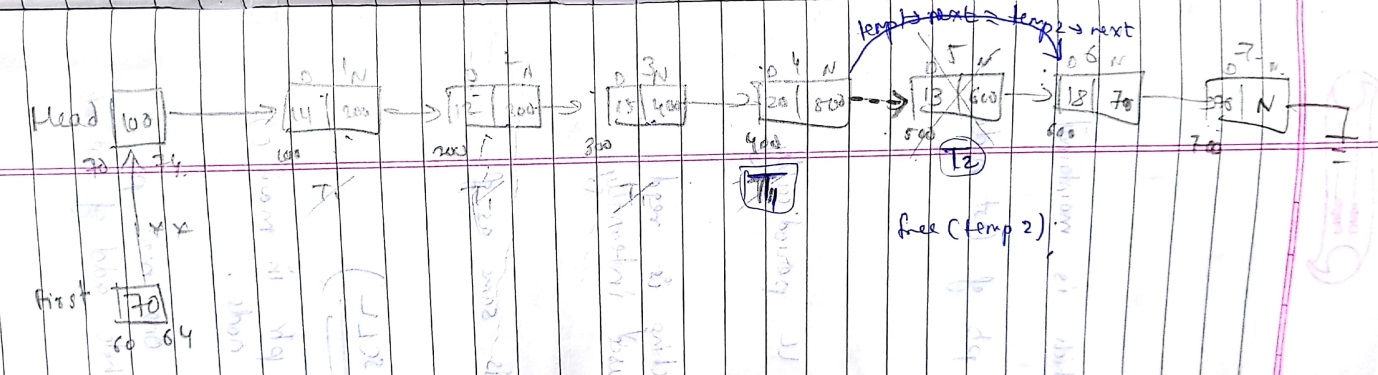
}

return TRUE;

}

//////////////////////////////////////////////////////////////////////////////

// Delete the node of specified position



BOOL DeleteAtPosition(struct node \*\*first,int pos)

{

struct node \*temp1 = \*first,\*temp2 = NULL;

if(pos <= 0)

{

return FALSE;

}

if(pos > (CountNode(\*first)))

{

return FALSE;

}

else if(pos == 1)

{

DeleteFirst(First);

}

else if(pos == (CountNode(\*First)))

{

DeleteLast(First);

}

else

{

While(pos!=2)

{

Temp1 = temp1 -> Next;

Pos--;

}

Temp2 = temp1 -> Next;

Temp1->next=temp2->next;

free( temp2 );

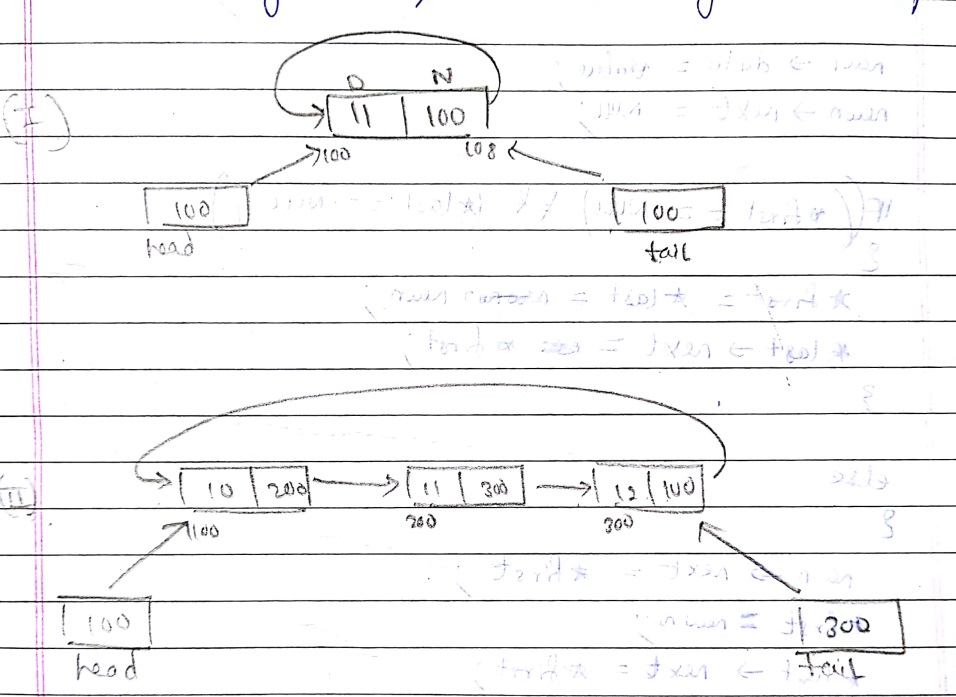
}

return TRUE;

}

/////////////////////////////////////////////////////////////////////////////////////////////

SINGLY CIRCULAR LINKED LIST



int InsertFirst (struct node \*\*First,struct node \*\*Last, int value)

{

struct node \*newn=NULL;

newn = (struct node \*)malloc(sizeof(struct node));

newn -> Data = value;

newn -> next = NULL;

if ((\*First == NULL) == && (\*Last == NULL))

{

\*First=\*Last = newn;

\*First = newn;

(\*Last) -> next = \*First;

}

else

{

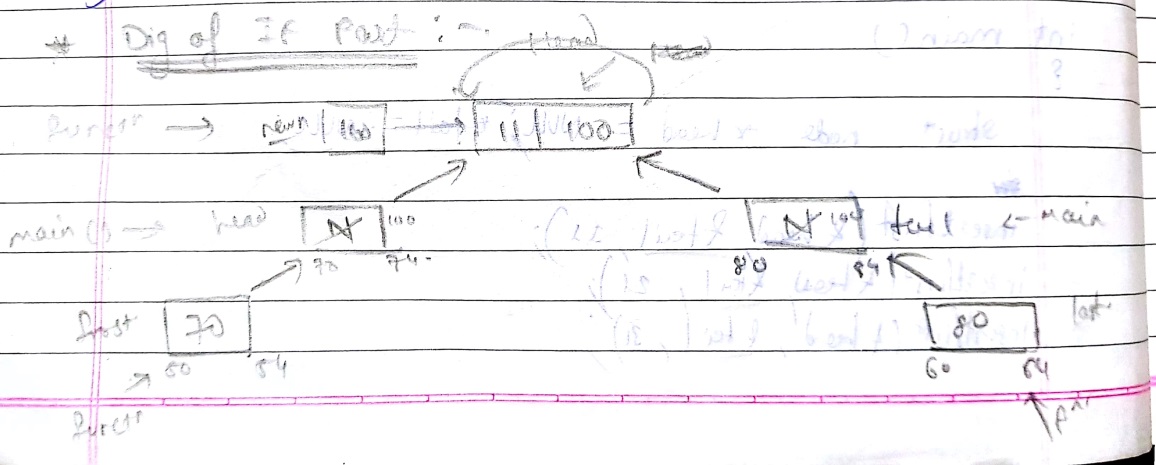
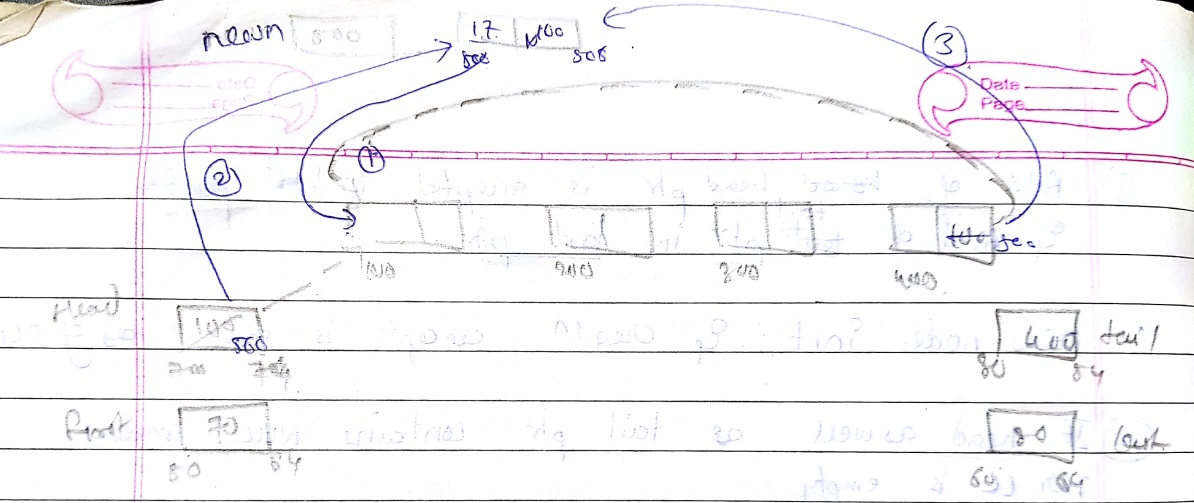
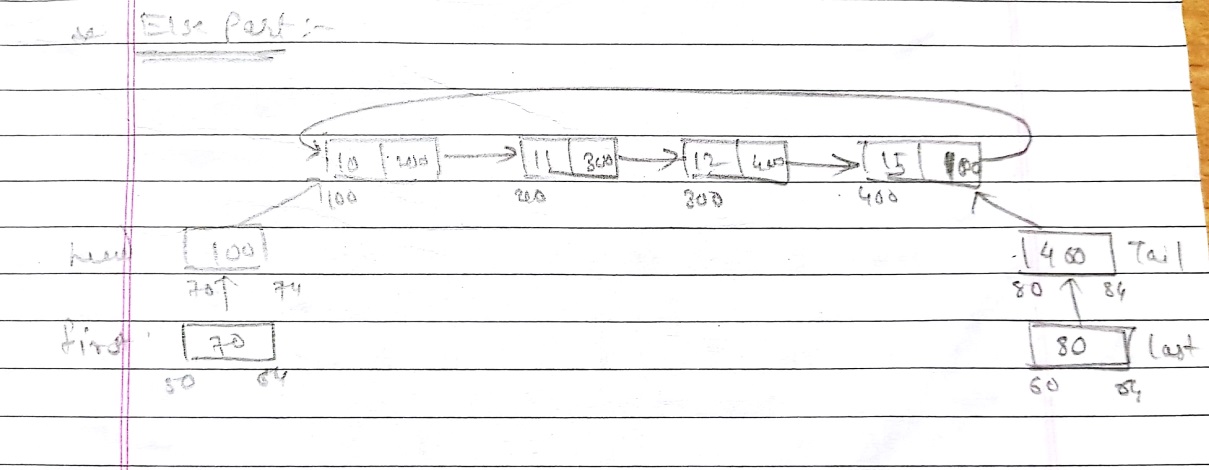
Newn -> next = \*First;

\*First = Newn;

(\*Last) ->next = \*first;

}

return TRUE;

} 

BOOL InsertLast (struct node \*\*First,struct node \*\*Last,int value)

{

struct node \*Newn=NULL;

Newn = (struct node \*)malloc(sizeof(struct node));

Newn -> Data = value;

Newn -> next = NULL;

if (\*First == NULL && \*Last == NULL)

{

\*First =\*Last= Newn;

(\*Last) -> next = \*First;

}

else

{

(\*Last) -> next = Newn;

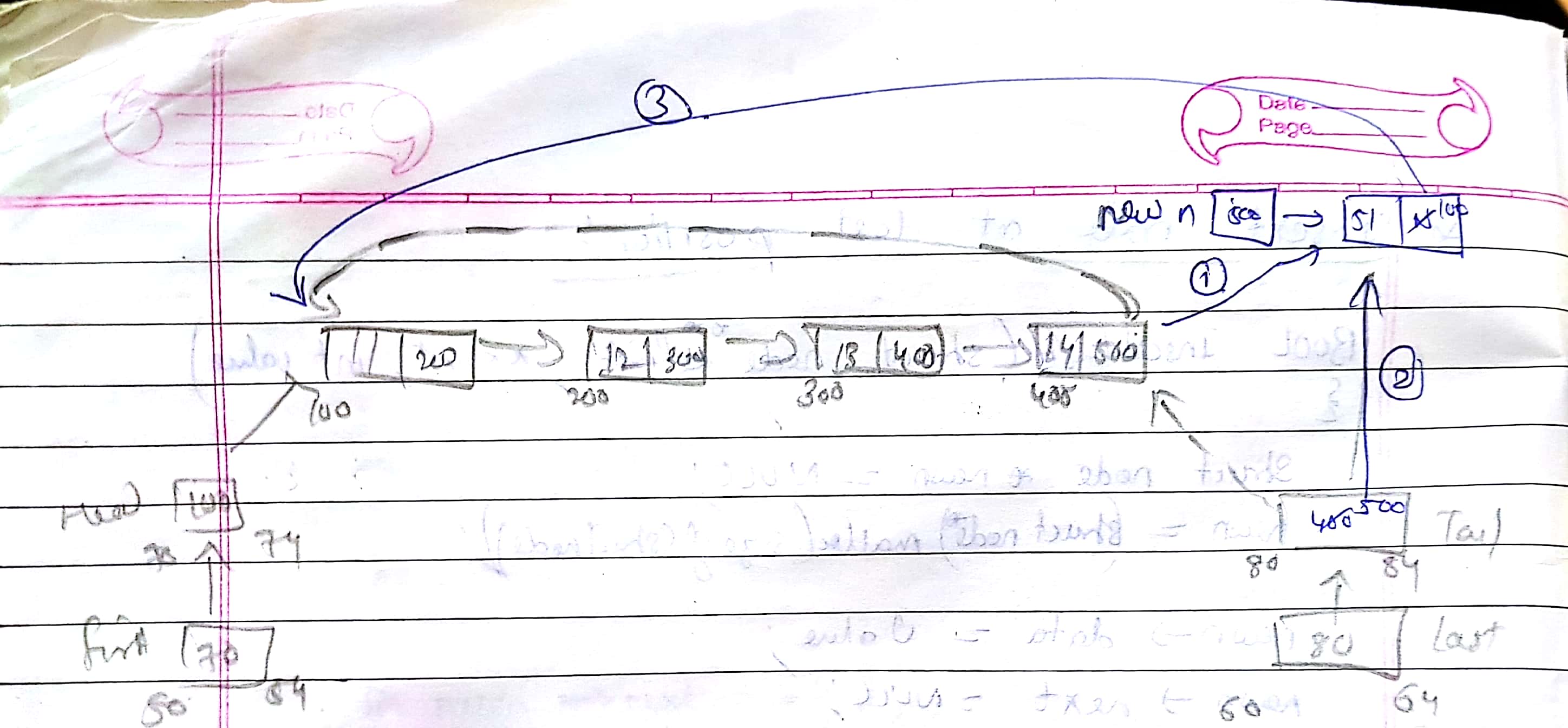
\*Last = Newn;

(\*Last) -> next = \*First;

}

return TRUE;

}



////////////////////////////////////////////////////////////////////////////

//Delete first node of SCLL

int DeleteFirst(struct node \*\*First,struct node \*\*Last )

{

if(\*First == NULL)&&(\*Last == NULL)

{

return FALSE;

}

elseif(\*First == \*Last)

{

Free(\*first);

\*first=\*last=NULL;

}

else

{

(\*First) = \*first -> next;

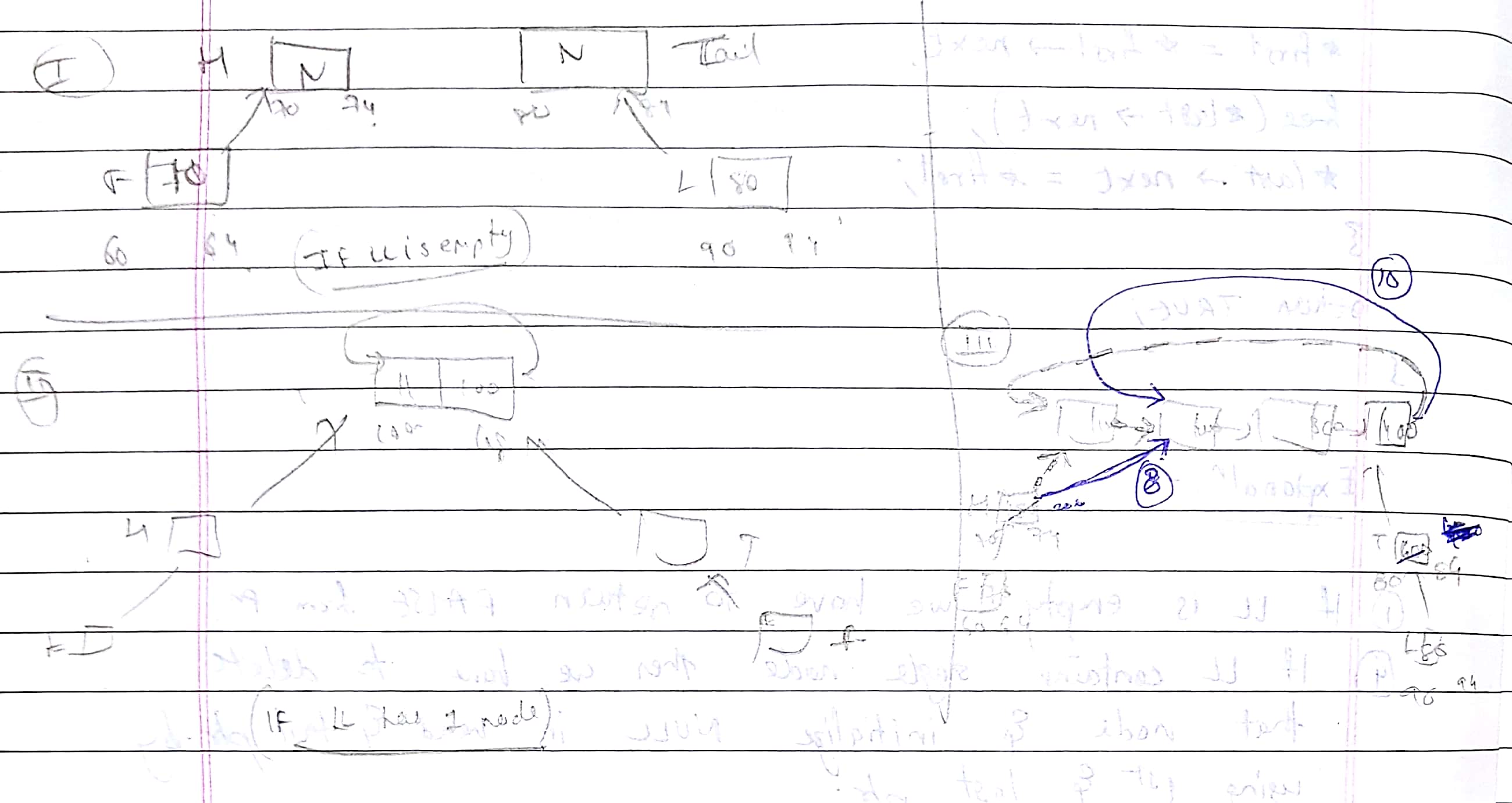
free(\*last->next);

\*last->next =\*first;

}

return TRUE;

}



///////////////////////////////////////////////////

//Delete last node

int DeleteFromLast(struct node \*\* first,struct node \*\* Last)

{

struct node \*temp = \*First;

if(\*First == NULL && \*Last == NULL)

{

return FALSE;

}

Elseif(\*first==\*last)

{

Free(\*last);

\*first=\*last=NULL;

}

else

{

while(temp -> next != \*last)

{

temp = temp -> next;

}

free(\*last);

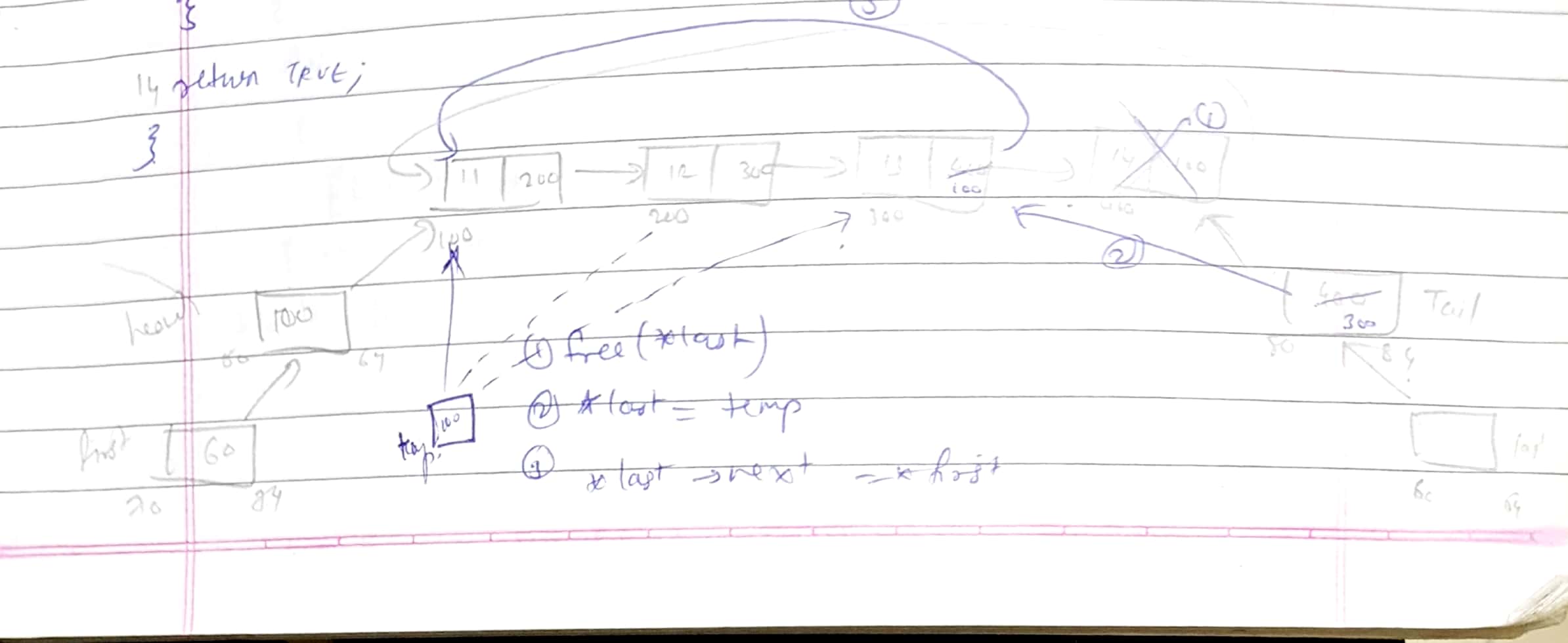
\*Last = temp;

\*last -> next = \*first;

}

return TRUE;

}



//Display the SCLL

void Display(struct node \*first,struct node \*last)

{

If((first!=NULL)&&(last!=NULL))

{

do

{

printf("|%d| -> ", first ->data);

first = first -> next;

}while(first != last->next);

}

}

///////////////////////////////////////////////////

int InsertAtPosition(struct node \*\* First,struct node \*\* Last, int pos, int no)

{

struct node \*temp = \*first;

if(pos<=0)

{

Return FALSE;

}

Else if( (pos > (CountNode(\*First,\*Last)+1))

{

return FALSE;

}

else if( pos == 1 )

{

insertFirst(First,Last,value);

}

else if(pos == (CountNode(\*First,\*Last)) + 1)

{

insertLast(First,Last,value);

}

struct node \*Newn = NULL;

Newn = (struct node \*)malloc(sizeof(struct node ));

if (newn ==NULL)

{

Return FALSE;

}

Newn -> Data = value;

Newn -> next = NULL;

while(pos!=2)

{

temp = temp -> next;

pos--;

}

Newn -> next = temp -> next;

temp -> next = Newn;

}

return TRUE;

}

////////////////////////////////////////////////////////////////

//delete at any specified position

BOOL DeleteAtPosition(struct node \*\* First,struct node \*\*Last,int pos)

{

Struct node \*temp1=\*first;

Struct node \*temp2=NULL;

If(pos<=0)

{

Return FALSE;

}

if(pos > (CountNode(\*First,\*Last)))

{

return FALSE;

}

else if(pos == 1)

{

DeleteFirst(First,Last);

}

else if(pos == (CountNode(\*First,\*Last)))

{

DeleteLast(First,Last);

}

else

{

While(pos!=2)

{

temp = temp -> next;

pos--;

}

Temp2 = temp1 -> next;

Temp1 -> next = temp2->next;

free( temp2);

}

return TRUE;

}