**Program:**

**/\* SLL DELETION \*/**

#include<stdio.h>

#include<stdlib.h>

void creation();

void traversal();

void deleting();

struct node

{

int data;

struct node \*link;

}\*ptr,\*header,\*new1,\*ptr1;

void main()

{

int ch;

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

header->link =NULL;

while(1)

{

printf("\n Enter the choice of operation 1.creation 2. delete 3.traversal : ");

scanf("%d",&ch);

switch(ch)

{

case 1: creation();

break;

case 2: deleting();

break;

case 3: traversal();

break;

default: exit(0);

}

}

}

void creation()

{

int x;

printf("enter the data value to insert");

scanf("%d",&x);

new1=(struct node \*)malloc(sizeof(struct node));

if(header->link==NULL)

{

header->link=new1;

new1->link=NULL;

new1->data=x;

}

else

{

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=new1;

new1->link=NULL;

new1->data=x;

}

}

void deleting()

{

int pos,x,key;

printf("\nEnter the position for deletion");

printf("\n 1.Begining 2.Ending\t3.At any Position\n");

scanf("%d",&pos);

if(pos==1) /\*Deletion at beginning\*/

{

ptr=header;

if(ptr->link==NULL)

{

printf("\n SLL is empty");

}

else

{

ptr1=ptr;

ptr=ptr->link;

ptr1->link=ptr->link; /\* Address of second node i.e link part of first node is copied to link part of header node\*/

printf("\nNode deleted is %d",ptr->data);

free(ptr);

}

}

else if (pos==2) /\* Deletion at Ending \*/

{

ptr=header;

if(ptr->link==NULL)

{

printf("\n SLL is empty, unable to perform deletion");

}

else

{

ptr1=ptr;

ptr=ptr->link;

while(ptr->link!=NULL)

{

ptr1=ptr;

ptr=ptr->link;

}

ptr1->link=NULL; /\* Last but one node link part is replaced with NULL \*/

printf("\nDeleted Node is %d",ptr->data);

free(ptr);

}

}

else if(pos==3) /\* Deletion at any position \*/

{

ptr=header;

if(ptr->link==NULL)

{

printf("\n SLL is empty, unable to perform deletion");

}

else

{

printf("\nEnter the data value to delete");

scanf("%d",&key);

while(ptr->data!=key && ptr->link!=NULL)

{

ptr1=ptr;

ptr=ptr->link; /\* Move to the next node\*/

}

if(ptr->link==NULL)

{

printf("\n Node with key was not found");

}

else

{

ptr1->link=ptr->link;

printf("\nDeleted node is %d",ptr->data);

free(ptr);

}

}

}

}

void traversal()

{

printf("\nelements in the list are");

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

printf("\t%d",ptr->data);

}

}

OUTPUT:

Enter the choice of operation 1.creation 2. delete 3.traversal : 2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

1

SLL is empty

Enter the choice of operation 1.creation 2. delete 3.traversal : 3

elements in the list are

Enter the choice of operation 1.creation 2. delete 3.traversal : 1

enter the data value to insert1

Enter the choice of operation 1.creation 2. delete 3.traversal : 1

enter the data value to insert2

Enter the choice of operation 1.creation 2. delete 3.traversal : 1

enter the data value to insert3

Enter the choice of operation 1.creation 2. delete 3.traversal : 1

enter the data value to insert4

Enter the choice of operation 1.creation 2. delete 3.traversal : 3

elements in the list are 1 2 3 4

Enter the choice of operation 1.creation 2. delete 3.traversal : 2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

1

Node deleted is 1

Enter the choice of operation 1.creation 2. delete 3.traversal : 3

elements in the list are 2 3 4

Enter the choice of operation 1.creation 2. delete 3.traversal : 2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

2

Deleted Node is 4

Enter the choice of operation 1.creation 2. delete 3.traversal : 3

elements in the list are 2 3

Enter the choice of operation 1.creation 2. delete 3.traversal : 2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

3

Enter the data value to delete2

Deleted node is 2

Enter the choice of operation 1.creation 2. delete 3.traversal : 3

elements in the list are 3