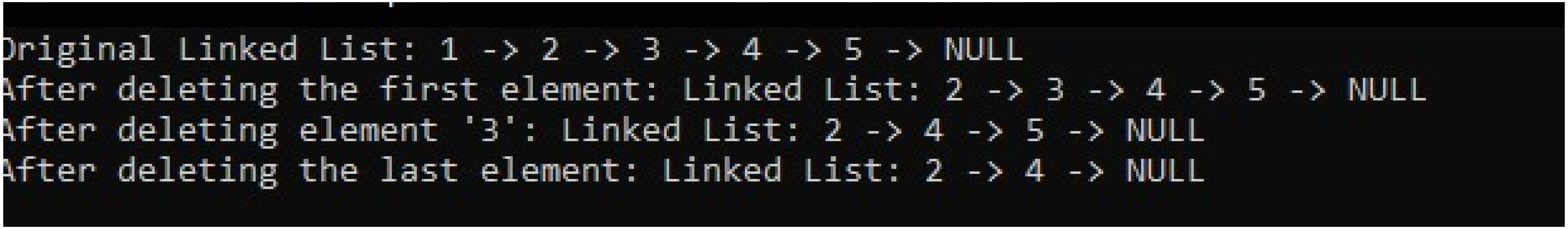

```
Original Linked List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
After deleting the first element: Linked List: 2 -> 3 -> 4 -> 5 -> NULL
After deleting element '3': Linked List: 2 -> 4 -> 5 -> NULL
After deleting the last element: Linked List: 2 -> 4 -> NULL
```

o/p: Original Linked List : 1→2→3→4→5→NULL.
After deleting the first element:
Linked List : 2→3→4→5→NULL
After deleting element '3' : Linked list : 2→4→5→NULL
After deleting the ~~latest~~ last element : Linked List : 2→4→NULL

22/01/24

e deleted");

```
else {
        while (ptr->data != val)
        {
            preptr = ptr;
            ptr = ptr->next;
        }
        preptr -> next = ptr->next;
        free(ptr);
        return start;
    }
};

Struct node *delete_after(struct node* Start)
{
 Struct node * ptr, *preptr;
 int val;
 Printf ("\n enter the value after which the node has to be deleted");
 Scanf ("%d", &val);
 ptr = start;
 Preptr = ptr;
 While (preptr->data != val)
 {
    preptr = ptr;
    ptr => ~~ptr->~~next;
 }
preptr->next = ptr->next;
free(ptr);
return start;
};
```

o/p: Original Linked
After deleting the
Linked List : 2-
After deleting ele
After deleting the

22/01/24

to be inserted:");

node));

[deletion]

```
Q2) Struct node * delete_beg (struct node * Start)
{
      Struct node * ptr};
      ptr = Start;
      Start = Star->next;
      free (ptr);
      return start;
};

Struct node * delete_end (struct node * Start)
{
      Struct node * Ptr, * ptreptr;
      Ptr= Start;
      While (ptr->next ! =Null)
      {
           Preptr=ptr;
           Ptr= ptr->next;
      }
      Pretptr->next= NULL;
      free (Ptr);
      Return Start;
};

Struct node * delete_node (struct node * Start)
{
      Struct node *ptr, *preptr;
      int Val;
      printf ("%d",&Val);
      Ptr =start;
      if (ptr->data= = Val)
      {
           Start = delete_beg (start);
           return= start;
      }
```

```
        New_node -> data = num;
        ptr = start;
        While (ptr->data != val)
        {
            ~~ptr~~preptr = ptr
            ptr = ptr->next;
        }.
        preptr->next = new_node;
        New_node->next = ptr;
        return start;
};

Struct node * insert_after (struct node * start)
{
    struct node * new_node, *ptr, * preptr;
    int num, val;
    Printf ("\n enter the data:");
    Scanf ("%d ", &num);
    Printf ("\n enter the value after which data has to be inserted:");
    Scanf ("%d", &val);
    New_node = (struct node *) malloc (sizeof (struct node));
    New_node -> data = num;
    ptr = start;
    Preptr = ptr;
    While (~~pre~~preptr->data != val)
    {
        Preptr = ptr;
        ptr = ptr->next;
    }
    ~~ptr~~preptr->next = new_node;
    New_node->next = ptr;
    return start;
};
```

[deletion]

```
Q2) Struct noc
    {
        Struct
        ptr =
        Start =
        free (p
        return
    };

    Struct node
    {
        Struct
        ptr = s
        While
        {
            Pre
            Ptr
        }
        Preptr
        free (pt
        return s
    };

    Struct node
    {
        Struct
        ~~int val~~
        Printf (
        ptr = st
        if (ptr
        {
            Start
            ret
        }
```

```
Struct node * insert_end(struct node * Start)
{
    Startet node *new_node, *ptr;
    int num;
    printf ("\n enter the data :");
    Scanf ("%d", &num);
    new_node = (struct node *) malloc (sizeof (struct node));
    New_node ->data = num;
    New_node -> next = NULL;
    Ptr = Start;
    if (start == NULL)
    {
        Start = new_node;
    }
    else
    {
    While (ptr->next != NULL)
            Ptr = ptr->next;
     ptr => next = new_node;
     return start;
     }
};

Struct node *insert_before(struct node * Start)
{
   Struct node * new_node, *ptr, *preptr;
   int num, val;
   printf("\n enter the data :");
   Scanf ("%d", &num);
   printf ("\n enter the value before which the data has to be inserted :");
   scanf ("%d", &val);
   new_node = (struct node *) malloc (sizeof (struct node));
```

```
          While(Ptr->next!=NULL)
                 ptr=Ptr->next;
          Ptr->next=new_node;
          new_node->next=NULL;
        }
        Printf ("\n enter the data:");
        Scanf ("%d", &num);
  }
  return start;
};

Struct node *display(struct node * Start)
{
   Struct node * ptr;
   Ptr = Start;
   While (ptr!=NULL)
   {
       printf ("\t %d", ptr->data);
       ptr = Ptr->next;
   }
};

Struct node * insert_beg(struct node * Start)
{
 Struct node * new_node;
int num;
printf ("\n enter the data:");
Scanf ("%d",&num);
New_node = (Struct node*)malloc (size of (struct node));
New_node -> data=num;
New_node -> next= start;
start = new_node;
return start;
};
```

```
Struct no
{
  Startet
  int nu
  printf
  Scanf ("
  new_no
  New_n
  New_n
  Ptr=
  if (st
  {
     S
  }
  else
  {
   While

    Ptr->
    return
   }
};

Struct node
{
  Struct nod
  int num,v
  printf("
  Scanf ("
  printf ("
  scanf ("
  New_node
```

f a node at first

```
e*);
e*);
de*).
ode*);
ode*);
de*);

*);

list \t2. display\t3. insert_beg
t_after");

created");
```

```
Case 2:  Start = insert_beg(start);
         break;
Case 3:  Start = insert_end(start);
         break;
Case 4:  Start = insert_before(start);
         break;
Case 5 :  Start = insert_after(start)
         break;
 }
}while (choice != 6);
return 0;
}

Struct node *create_ll (struct node *start)
{
  Struct node  * new_node, *ptr;
  int num;
  printf ("\n enter -1 to end ");
  printf (" \n enter the data: ");
  Scanf ("%d", &num);
  While (num != -1)
  {
     New_node = (struct node *) malloc (size of(struct node));
     New_node -> data = num;
     if (start == NULL)
     {
        New_node -> next = NULL;
        Start = new_node;
     }
     else
     {
          Ptr = Start;
```

Singly Linked List

[Create a linked list and insertion of a node at first position]

```
#include<Stdio.h>
#include<Malloc.h>
Struct node
{
      int data;
      Struct node *next;
};
Struct node * Start=Null;
Struct node * Create_ll(Struct node*);
Struct node * display (Struct node*);
Struct node * insert_beg(Struct node*).
Struct node * insert_end(Struct node*);
Struct node * insert_before(struct node*);
Struct node * insert_after(struct node*);

Struct node * Sort_list(struct node*);
int main()
int Choice;
Printf("\n**menu**\n1.create a list\t2.display\t3.insert_beg
      \t4insert_end\t5 insert_before\t6 insert_after");
do
{
   printf("\n enter the Choice:");
   Scanf("%d, &Choice:);
   Switch(Choice)
   {
     Case 1: Start=Create_ll(start);
             Printf("\n linked list Created");
             break;
```

```
Case 2:  Start=
         break;
Case 3:  Start=
         break;
Case 4:  Start=
         break;
Case 5:  Start=
         break;
 }
}while(Choice!=6
return 0;
}

Struct node *Cr
{
  Struct node *n
  int num;
  printf("\n ente
  printf("\n ent
  Scanf("%d",&n
  While(num!=-1)
  {
     New_node=(st
     New_node->
     if(start==
     {
        New_node
        Start=
     }
     else
     {
        Ptr=
```