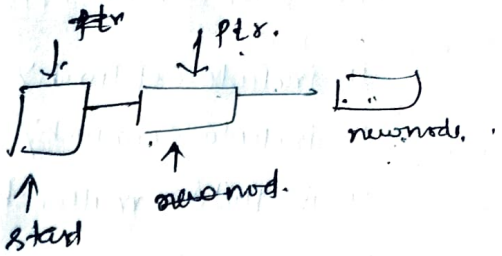
Create a Linked list

```
#Include <stdio.h>
#Include <conio.h>
#include <stdlib.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*)
int main()
   {
start= create_ll (start);
   }

 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 (sizeof (struct node));
            new-node -> data = num;
          if ( start == NULL)
             {
               new_node -> next = NULL;
                 start = new_node;
             }
           else
             {
```

```
        ptr = start
        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 ("%d", ptr -> data);
            ptr = ptr -> next;
        }
        return start;
```

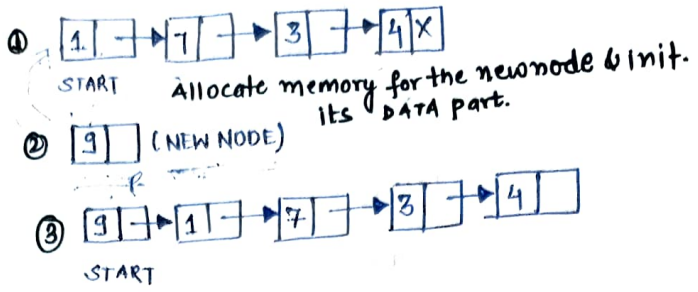


Insertion in a Linked List:

(a) Insertion @ Beginning of linked list.

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>

struct node
  {
    int data;
    struct node * next;
  };
```




```
struct node *start = NULL;
struct node * insert_beg (struct node*);

int main ( )
  {
start= insert_beg (start);
  }
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 (sizeof(struct node));
     new_node -> data = num;
     new_node -> next = start;
      start = new_node;
      return (start);
 }
```

```
ptr = start;
while (ptr -> next != start)
    ptr = ptr -> next;
ptr -> next = new_node
```

(5) Insertion at End.

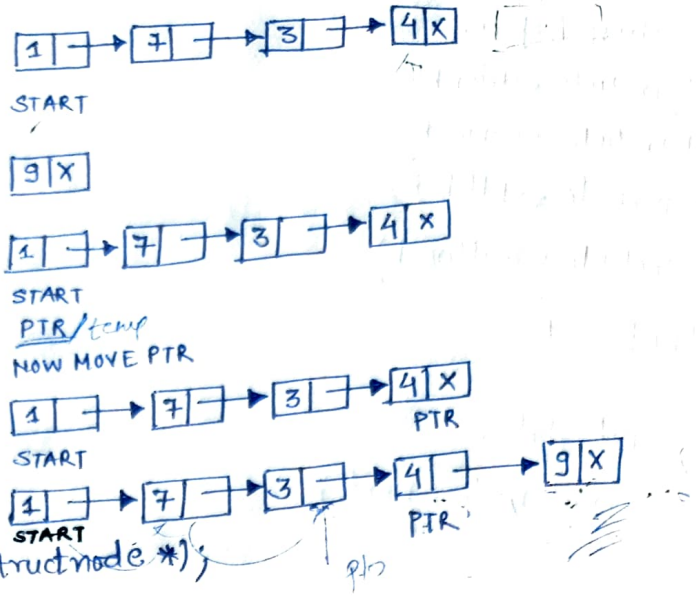
```
# include <stdio.h>
# include <conio.h>
# include <malloc.h>

Struct node
  {
  int data;
  struct node * next;
  };

Struct node * start = NULL;
Struct node * insert_end (struct node *);

int main()
{
start = insert_end (start);
}

struct node * insert_end (struct node *start)
  {
   struct node * ptr, * new_node;
   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;
     while (ptr -> next != NULL)
       ptr = ptr -> next;
       ptr -> next = new_node;
     return start;
  }
```



© Insertion Before.

```
#Include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
Struct node
  {
   int data;
   Struct node *next;
   };
   Struct node * stast= NULL;
   Struct node *insest_before (struct nolle*);

  int main()
    {
Start= insest_before(stast);
    }
Struct node *insest_before (struct node * stast)
   {
    struct node *new_node, *ptr, *preptr;
    int num, val
   printf("\n Enter the data");
   scanf ("%od", &num);
   printf ("\n Enter the value before which data insested:");
   scanf ("%od", &val);
   new_node = (struct node *) malloc (sizeof (struct node));
    new_node ->data = num;
    ptr = stast;
    while (ptr != val)
       {
        preptr = ptr;
        ptr = ptr->next;
       }
      preptr -> next = new_node;
      new_node -> next = ptr;
  } return start;
```

④ Insert after.

```
#include <stdio.h>
# include <conio.h>
# include <malloc.h>
# include<stdlib.h>
Struct node
  {
   int data;
   struct node *next;
  };
Struct node * start=NULL;
Struct node * insert_after (struct node*);

int main()
  {
   start= insert_after (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 the 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 ( preptr -> data != val)
        {
          preptr = ptr;
          ptr = ptr->next;
        }
       preptr -> next = new node;
       new node ->next = ptr;
        return start;
 }
```

(e) Deletion of Node at the beginning

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

(f) Deletion of Node at the end.

```
struct node * delete_end (struct node *start)
{
  struct node *ptr, *preptr;
   ptr = start;
   while (ptr->next != NULL)
        {
          preptr = ptr;
          ptr = ptr->next;
        }
       preptr->next = NULL;
        free (ptr);
        return start;
   }
```

(g) Delete after

```
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;
}
```

(h) Sorting of the list

```
struct node * sort_list(struct node *start)
  {
    struct node *ptr1, *ptr2;
    int temp;
     ptr1 = start;
     while (ptr1 -> next != NULL)
        {
          ptr2 = ptr1 -> next;
          while( ptr2 != NULL)
            {
              if ( ptr1 -> data > ptr2 -> data)
                  {
                    temp = ptr1 -> data;
                    ptr1 -> data = ptr2 -> data;
                    ptr2 -> data = temp;
                  }
                ptr2 = ptr2 -> next;
            }
          ptr1 = ptr1 -> next;
        }
        return start;
```