```c
#include<stdio.h>
#include<stdlib.h>
// ***************************
struct node
{
 int data;
 struct node*next;
 struct node* prev;
};
//*******************************
struct node*createnode()             //node creation
{
 int data;
 struct node*newnode = NULL;
 newnode = (struct node*)malloc(sizeof(struct node));
 if(newnode == NULL)
 {
 printf("Memory not allocated\n");
 return NULL;
 }
 else
 {
  printf("Enter the data\n");
  scanf("%d",&data);
  newnode->data = data;
  newnode->next = NULL;
  newnode->prev = NULL;
  return newnode;
 }
}
//*******************************
void create_linklist(struct node**head_first,struct node**head_last)          //
createlinklist
{
  struct node*newnode = NULL;
  struct node*travnode = *head_first;
  newnode = createnode();
  if(*head_first == NULL)
  {
   *head_first = newnode;
   *head_last  = newnode;

  }
  else
  {
   (*head_last)->next = newnode;
    newnode->prev = *head_last;
    *head_last = newnode;
 }
}
//*******************************

int countnode(struct node*head_first)        //countnode;
{
  int count = 0;
  while(head_first!=NULL)
  {
   count++;
   head_first = head_first->next;
  }
  return count;
}

//***************************
void insert_at_first(struct node**head_first,struct node**head_last)
{
```

```
  struct node*newnode = createnode();          //insert at 1st
   if(*head_first == NULL)
   {
     *head_first = newnode;
     *head_last  = newnode;

   }
   else
   {

   (*head_first)->prev =newnode;
    newnode->next =*head_first;
   *head_first = newnode;
   newnode->prev = NULL;
   }

//**********************
}
void insert_at_position(struct node**head_first,struct node**head_last)        //
insert at a position
{
 int pos,count;
 printf("Enter the position at which you want to insert a node\n");
 scanf("%d",&pos);
 count = countnode(*head_first);
 if(pos == 1)
 {
   insert_at_first(head_first,head_last);
 }else if(pos == count+1)
         {
           create_linklist(head_first,head_last);
         }
  else if(pos < 1 || pos > count+1)
         {
          printf("Invalid position to insert a node\n");
          insert_at_position(head_first,head_last);
         }

  else{
   struct node *newnode = NULL;
   newnode = createnode();
   struct node *tempnode = *head_first;
   struct node *tempnode1 = *head_last;
   if(pos <= count/2)
   {
     for(int i=1;i<(pos-1);i++)
     {
       tempnode = tempnode->next;
     }
     newnode->next = tempnode->next;
     tempnode->next = newnode;
     tempnode->next->prev = newnode;
     newnode->prev = tempnode;
   }
   else
   {
   for(int i =count;i>(pos);i--)
   {
     tempnode1 =tempnode1->prev;
   }
   newnode->next  = tempnode1;
   newnode->prev  = tempnode1->prev;

   tempnode1->prev->next =newnode;
   tempnode1->prev = newnode;
```

```
   }

  }


}
//*****************************
void display_linklist(struct node*head)
{                                                    //display linklist
 printf("Your linklist is \n");
 while(head!=NULL)
 {
  printf("%d\t",head->data);
  head = head->next;
 }
 printf("\n");
}
//*************************
void rev(struct node*head_last)

{                                                    // reverse linklist
  while(head_last!=NULL)
  {
   printf("%d\t",(head_last)->data);
   head_last =head_last->prev;
  }

 }

 //*****************************


 void delete_first(struct node**head_first)      //delete first node
{
  struct node * ptr = *head_first;
  *head_first = ptr->next;
  (*head_first)->prev = NULL;
  free(ptr);
}
//*****************************
void delete_last(struct node **head_last)
{
                                                        //delete last node
 struct node *ptr = *head_last;
 *head_last = ptr->prev;
 (*head_last)->next = NULL;
 free(ptr);
}
//*******************************
void delete_at_position(struct node**head_first,struct node**head_last)        //
delete at a position
{
 int pos,count;
 printf("Enter the position at which you want to delete a node\n");
 scanf("%d",&pos);
 count = countnode(*head_first);
 if(pos == 1)
 {
   delete_first(head_first);
 }else if(pos == count)
         {
           delete_last(head_last);
         }
  else if(pos < 1 || pos > count+1)
         {
          printf("Invalid position to deletea node\n");
```

```
          delete_at_position(head_first,head_last);
         }

  else{
   struct node *ptr = NULL;
   struct node *tempnode = *head_first;
   struct node *tempnode1 = *head_last;
   if(pos <= count/2)
   {
     for(int i=1;i<(pos-1);i++)
     {
       tempnode = tempnode->next;
     }
     ptr = tempnode->next;
     tempnode->next = ptr->next;
     ptr->next->prev = tempnode;
     free(ptr);

   }
   else
   {
   for(int i =count-1;i>(pos);i--)
   {
     tempnode1 =tempnode1->prev;
   }
    ptr = tempnode1->prev;
    tempnode1->prev = ptr->prev;
    ptr->prev->next = tempnode1;
    free(ptr);
   }

  }


}
//*****************************
void main()
{                                                  // main function
 struct node * first = NULL;
 struct node* last = NULL;


  int choice;
 do
 {
  printf("1. Create linklist\n");
  printf("2. Display linklist\n");
  printf("3. Reverse linklist\n");
  printf("4. Insert at first\n");
  printf("5. Insert at a position\n");
  printf("6. Delete first\n");
  printf("7. Delete last\n");
  printf("8. Delete at a position\n");
  printf("9. Exit\n");
  printf("*******************************\n");
  printf("Enter your choice\n");
  scanf("%d",&choice);
  switch(choice)
  {
   case 1: create_linklist(&first,&last);
           break;
   case 2: display_linklist(first);
           break;
   case 3: rev(last);
           break;
   case 4: insert_at_first(&first,&last);
```

```
            break;
    case 5: insert_at_position(&first,&last);
            break;
    case 6:delete_first(&first);
            break;
    case 7:delete_last(&last);
            break;
    case 8:delete_at_position(&first,&last);
            break;
   }
  }while(choice!=9);
}
```