```
#include<stdio.h>
#include<stdlib.h>

//structure definition
struct list
{
      int info;
      struct list *next;
};

//Functions Prototypes
void create(struct list **, int);
void traverse(struct list *);
void insert_first(struct list **,int);
void insert_last(struct list **, int);
int count(struct list *);
void delete_first(struct list **);
void delete_last(struct list **);
void reverse(struct list **);
void del_node(struct list **, int);
void insert_after(struct list *, int, int);
void insert_before(struct list **, int, int);

//Main Function
int main()
{
      int num, c, item,item2;
      struct list *head=NULL;
      while(1)
      {
            //switch to display menu
            printf("1.Create\n2.Traverse\n3.Insert First\n4.Insert
Last\n5.Delete First\n6.Delete Last\n7.Count\n8.Reverse\n9.Delete a
node with specific element\n10.Insert a element before certain
element\n11.Insert a element after certain element\n0.Exit\nYour
Choice: ");
            scanf("%d",&c);
            switch(c)
            {
                  case 1:
                        printf("\nEnter the number of nodes: ");
                        scanf("%d",&num);
                        create(&head, num);
```

```
                    break;

                case 2:
                        traverse(head);
                        break;
                case 3:
                        printf("\nEnter the information for the
node to be inserted: ");
                        scanf("%d",&item);
                        insert_first(&head,item);
                        break;

                case 4:
                        printf("\nEnter the information for the
node to be inserted: ");
                        scanf("%d",&item);
                        insert_last(&head,item);
                        break;

                case 5:
                        delete_first(&head);
                        break;

                case 6:
                        delete_last(&head);
                        break;

                case 7:
                     printf("\nNumber of nodes: %d", count(head));
                     break;

                case 8:
                     reverse(&head);
                     break;

                case 9:
                     printf("\nEnter the element of node to delete:
");
                     scanf("%d",&item);
                     del_node(&head,item);
                     break;

                case 10:
```

```
                        printf("\nEnter the element of node after
which you want to insert: ");
                        scanf("%d",&item);
                        printf("\nEnter the value to insert: ");
                        scanf("%d",&item2);
                        insert_before(&head,item,item2);
                             break;

                case 11:
                        printf("\nEnter the element of node after
which you want to insert: ");
                             scanf("%d",&item);
                       printf("\nEnter the value to insert: ");
                       scanf("%d",&item2);
                       insert_after(head,item,item2);
                             break;

                case 0: exit(0);
                default:
                        printf("\nWrong input. Please try
again...");
           }
      }
      return(0);
}

//Function definition to create linked list
void create(struct list **phead, int num)
{
      struct list *temp,*newnode;
      int item,i;
      if(*phead != NULL)
      {
                printf("Already created");
                return;
      }
      for(i=1;i<=num;i++)
      {
            printf("Enter the information to be stored in a node:
");
            scanf("%d",&item);
            newnode=(struct list *)malloc(sizeof(struct list));
            newnode->info=item;
```

```
            newnode->next=NULL;

            if(*phead == NULL)
                  *phead=newnode;
            else
                  temp->next=newnode;

            temp=newnode;
      }
      return;

}

//Function definition to display linked list
void traverse(struct list *head)
{
      struct list *loc;
      loc=head;

      while(loc!=NULL)
      {
            printf("%d ",loc->info);
            loc=loc->next;
      }
      printf("\n");
}

//Function definition to insert element in first place
void insert_first(struct list **phead, int item)
{
      struct list *newnode;
      newnode = (struct list *) malloc(sizeof(struct list));
      newnode->info = item;
      newnode->next = *phead;
      *phead = newnode;
      return;
}

//Function definition to insert element in last place
void insert_last(struct list **head, int item)
{
      struct list  *loc, *newnode;
      newnode = (struct list *) malloc(sizeof(struct list));
```

```
      newnode->info = item;
      newnode->next = NULL;
      loc=*head;

      while(loc->next!=NULL)
      {
            loc=loc->next;
      }

      loc->next = newnode;
      return;
}

//Function definition to count element in linked list
int count(struct list *head)
{
      int count=0;
      struct list *loc;
      loc=head;
      while(loc!=NULL)
      {
            count+=1;
            loc=loc->next;
      }
      return count;
}


//Function definition to delete element in first place
void delete_first(struct list **phead)
{
      struct list *temp;

      if(*phead == NULL)
      {
            printf("\nEmpty List...Deletion is impossible....");
            return;
      }

      temp = *phead;
      *phead = (*phead)->next;

      printf("\nInformation on deleted note is %d\n",temp->info);
```

```
	temp->next = NULL;
	free(temp);

	return;
}

//Function definition to delete element in last place
void delete_last(struct list **phead)
{
	struct list *loc, *locp;
	if(*phead==NULL)
	{
		printf("\nEmpty List");
		return;
	}

	loc=*phead;
	locp=NULL;

	while(loc->next!=NULL)
	{
		locp=loc;
		loc=loc->next;
	}

	printf("\nInformation on deleted node is %d\n",loc->info);

	if(loc==*phead)
		*phead=loc->next;
	else
		locp->next=loc->next;

	free(loc);
}

//Function definition to reverse a linked list
void reverse(struct list **phead)
{
	struct list *locp, *loc, *locn;
	if(*phead == NULL || (*phead)->next == NULL)
	{
```

```
      printf("\nEither Empty List or List contains EXACTLY one
node....");
      return;
      }
      loc = *phead;
      locp = NULL;

      while(loc != NULL)
      {
      locn = loc->next;
      loc->next = locp;
      locp = loc;
      loc = locn;
      }

      *phead = locp;
      return;
}
```

//Function definition to delete a particular element from the linked list

```
void del_node(struct list **phead, int item)
{
      struct list *loc, *locp;
      if(*phead == NULL)
      {
            printf("\nEmpty List ....So deletion is
impossible....");
            return;
      }

      loc=*phead;
      while(loc != NULL && loc->info != item)
      {
            locp = loc;
            loc = loc->next;
      }

      if(loc == NULL)
      {
            printf("\nNode to be deleted is not found...");
            return;
      }
```

```
      if(loc == *phead)
            *phead = loc->next;
      else
            locp->next = loc->next;

      loc->next = NULL;
      free(loc);
      return;
}

//Function definition to insert an element before a particular
element
void insert_before(struct list **phead, int item1, int item2)
{
      struct list *new_node = NULL;
      struct list *tmp = *phead;
      new_node = (struct list *)malloc(sizeof(struct list));

      if (new_node == NULL)
      {
         printf("Failed to insert element. Out of memory");
         return;
      }

      new_node->info = item2;
      if ((*phead)->info == item1)
      {
         new_node->next = *phead;
         *phead = new_node;
         return;
      }

      while (tmp && tmp->next)
      {
         if (tmp->next->info == item1)
         {
             new_node->next = tmp->next;
             tmp->next = new_node;
             return;
         }
         tmp = tmp->next;
      }
```

```
      /*Before node not found*/
      free(new_node);
}

//Function definition to insert element after a particular element
void insert_after(struct list *head, int item1, int item2)
{
      struct list *new_node = NULL;
      struct list *tmp = head;

      while(tmp)
      {
            if(tmp->info == item1)
            {
                  // found the node
                  new_node = (struct list *)malloc(sizeof(struct
list));

                  if (new_node == NULL)
                  {
                        printf("Failed to insert element. Out of
memory");
                  }

                  new_node->info = item2;
                  new_node->next = tmp->next;
                  tmp->next = new_node;
                  return;
            }
            tmp = tmp->next;
      }
}
```