1BM19CS054
Dinyashree.K

Lab 5

```
#include<stdio.h>
#include<stdlib.h>
void create();
void display();
void insert_at_node(int);
void insert_before();
struct node
{
  int data;
  struct node* next;
};
struct node* head=NULL;
int main(int argc, char* *argv)
{
  int choice, ch;
  do{
  printf("\n1. Create \n2. Display \n3.
        Insert_before \n4. Insert at particular
        node \n5. Exit\n");
  printf("\n Enter your choice : ");
  scanf("%d",&choice);
  switch(choice)
  {
  case1: create(); break;
  case2: display(); break;
  case3: insert_before(); break;
  case4: printf("Enter the node where the new
                element has to be inserted \n");
         scanf("%d",&ele);
         insert_at_node(ele);
         break;
  default: exit(0);
  }
} while(choice==1||choice==2||choice==3||
        choice==4);
}
```

```
void create()
{
 struct node * newnode, * temp;
 int item;
 newnode = (struct node *) malloc (size of
                                   (struct node));

 printf("\n Enter the data: ");
 scanf("%d", &item);
 newnode -> data = item;
 if (head == NULL)
 {
  newnode -> next = NULL;
  head = newnode;
  printf("\n Node created \n");
 } else
 {
  temp = head; // transversing
  while (temp -> next != NULL)
  {
   temp = temp -> next
  }
  temp -> next = newnode;
  newnode -> next = NULL;
  printf("\n Node created \n");
 }
}
void insert_before()
{
 struct node * newnode;
 int ele;
 printf("\n Enter the element: ");
 scanf("%d", &ele);
 newnode = (struct node *) malloc (size of
                                   (struct node));

 newnode -> data = ele;
 newnode -> next = head;
  head = newnode;
```

```
}
void insert_at_node(int a)
{
  struct node* newnode,*temp;
  int i, ele;
  printf("Enter the element:");
  scanf("%d",&ele);
  temp = head;
  newnode=(struct node*)malloc(sizeof
                    (struct node));
  newnode->data=ele;
  while(i<0)
  {
     temp = temp->next;
     i++;
  } newnode->next = temp->next;
  temp->next = newnode;
}
void display()
{ struct node *ptr=NULL;
   ptr = head;
   if(ptr==NULL){
   printf("List empty !!!\n"); }
   else
   { while(ptr!=NULL)
      { printf("%d", ptr->data);
         ptr = ptr->next;
      }
   }
   printf("\n");
}
```