23.11.20
Aniket Prasad
1BM19CS194

Lab 6- Implementation of singly linked list -

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head;

void beginsert()
 {
    struct node *ptr;
    int item;
    ptr = (struct node*) malloc (sizeof (struct node*));
     if (ptr = null)
    {
      printf ("\n overflow");
  }
   else
   {
      ptr -> data = item;
      ptr -> next = head;
     head = ptr;
      printf ("\n Node inserted");
    }
 }
 void lastinsert ()
{
```

```
ptr = (struct node*) malloc (sizeof (struct node));
 if (ptr == NULL)
{
    printf ("\n OVERFLOW");
}
 else
{
    printf  ptr->data = item;
          if (head == NULL)
      {
           ptr -> next = NULL;
           head = ptr;
           printf ("\n NODE inserted");
      }
       else
      {
         temp = head;
         while (temp -> next != NULL)
        {
           temp = temp -> next;
      }
       temp ->next=ptr;
        ptr -> next = NULL;
      printf ("\n Node Inserted");
    }
  }
}

void randominsert ()
{
   ptr = (struct node *) malloc (size of (struct node));
   if (ptr == NULL)
```

```
{
    printf ("\n OVERFLOW");
}
else
{
    ptr -> data = item ;
    printf("\n Enter the loc
    scanf()
    temp = head ;
    for (i = 0; i < loc ;i++ )
{
      temp = head; temp -> next ;
      if (temp == NULL)
      {
          printf ("\n Cant insert \n");
          return ;
      }
}
      ptr -> next = temp -> next ;
      temp -> next = ptr ;
    }
}
void begin_delete ()
{
      struct node * ptr ;
      if (head == NULL)
    {
        printf("\n List is empty \n");
    }
    else
    {
```

```
    ptr = head;
    head = ptr -> next;
    free(ptr);
    printf("\n");
  }
}

void last_delete()
{
  struct node *ptr, *ptr1;
  if (head == NULL)
  {
    printf("\n");
  }
  else if (head -> next == NULL)
  {
    head = NULL;
    free(head);
  }
  else
  {
    ptr = head;
    while (ptr -> next != NULL)
    {
      ptr1 = ptr;
      ptr = ptr -> next;
    }
    ptr1 -> next = NULL;
    free(ptr);
    printf("\n");
  }
}
```

```
void random_delete ()
{
    struct node * ptr, * ptr1;
    int loc, i;
    ptr = head;
    for (i=0; i<loc; i++)
{
    ptr1 = ptr;
    ptr = ptr->next;
    if (ptr == NULL)
  {
      ptr printf ("\n Can't delete");
  }
}
ptr1->next = ptr->next;
free (ptr);
}

void search ()
{
    if (ptr == NULL)
{
    printf ("Empty list");
}
  else
  {
    search item;
  while (ptr1 = NULL)
{
    if (ptr->data == item)
  {
```

```
}
 else
{
      flag = 1;
}
i++
  ptr = ptr -> next;
}
 if (flag == 1)
{
  printf("    ");
 }
}
}
 void display()
{
  if ptr (ptr == NULL)
   {
      print nothing ;
   }
   else
    {
     "values are :- " ;
   while (ptr != NULL)
     {
        ptr = ptr -> next;
     }
    }
  }
```