Lab 6.c

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

   void insert_beg();
   void insert_last();
   void insert_random();
   void delete_beg();
   void delete_last();
   void display();
   int  item;

  void main() {
         int choice = 0;
         while(1) {

        printf(" **MENU** \n");
        printf("Choose your option\t" );
        printf(" [1] Insert at Begining \n");
        printf(" [2] Insert at Last \n");
        printf(" [3] Inbetrt at Random \n");
        printf(" [4] Delete at Begining \n");
        printf(" [5] Delete at Last \n");
        printf(" [6] Delete at Random \n");
        printf(" [7] Display \n");
        printf("[8] Exit...\n");
printf("Enter your option: \n");
scanf("%d",&choice);
switch (choice) {

    case 1:
          insert_beg();
          break;
    case 2:
          insert_last();
          break;
    case 3:
          insert_random();
          break;
    case 4:
          delete_beg();
          break;
    case 5:
          delete_randomLast();
          break;
    case 6:
          delete_random();
          break;
    case 7:
          display();
          break;
    case 8:
          exit(0);
          break;

      default: printf("\n Invalid Option!! \n");
      }
   }
}
```

```
void insert_beg() {

    struct node *ptr;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
      {
        printf("\n Overflow*\n");
      }

    else
      {
        printf(" Enter value of node ::>\n");
        scanf("%d", &item);
        ~~printf("~~ ptr->data = item;
              ptr -> next = head;
                head = ptr;
              printf(" Node has been Successfully
                    Inserted !!\n");

          }
        }
void insert_last() {

      struct node *ptr, *temp;
      ptr = (struct node *) malloc(sizeof (struct node));
      if(ptr == NULL)
           {
             printf("\n Overflow!!!\n");
           }
        else {
             printf(" Enter value of Node ::> \n");
              scanf("%d", &item);
              ptr -> data = item;
if (head == NULL)
  {
     ptr -> next = NULL;
     head = ptr;
     printf(" Node successfully inserted!!!\n");
     }
  else
    {
       temp = head;
       while (temp -> next! == NULL);
          {
            temp = temp -> next ;
          }
          temp -> next = ptr ;
          ptr -> next = NULL;
          printf(" Node successfully Inserted!!\n");
          }
       }
  }

void insert_random( ){
      int locat;
      struct node *ptr, *temp ;
      ptr = (struct node*) malloc (sizeof(struct node));
      if (ptr == NULL){
                printf ("\n Overflow!!"\n");
           }
```

```
else
{   printf (" Enter the value of the Node ::>\n");
    scanf (" %d", &item);
    ptr -> data = item;
    printf (" Enter the location to which you want the element
     ~~scanf~~(" to be inserted ::>\n");
     scanf (" %d", &locat);
     temp = head;
     if (locat == 1){

          ptr -> next = temp;
           head = ptr;
          return;

     }
     for (int i = 0; i < locat - 1; i++){
            temp = temp -> next;
           if ( temp = NULL){
                printf ("\nInsertion failed !! \n");
                return;
             }
          }
         ptr -> next = temp -> next;
            temp -> next = ptr;
           printf (" Insertion Successfully\n");
           }
      }
```

```
void dilete_ last() {
  struct node *ptr, *ptr1;
    if (head == NULL)
        {
          printf("\n List is empty !!! \n");
        }
 else if (head -> next == NULL) {
          head = NULL;
          free(head);
          printf(" Only Node of List Deletion !!! \n");
        }
     else
      {
        ptr = head;
       while (ptr->next != NULL) {
            ptr1 = ptr;
             ptr = ptr -> next;
            }

       ptr1 -> next = NULL;
          free(ptr);
         printf ("\n Deleted Node from Last\n");
          }
      }
  }
```

```
void delete_random (){

        struct node *ptr, *ptr1 ;
        int locat ;
        printf ("Delete location of Node to be deleted\n");
         scanf ("%d",&locat);
         ptr = head;
           for (int i=0; i<locat ; i++) {

                   ptr1 = ptr ;
                  ptr = ptr -> next;
                  if(ptr ==NULL){
                          printf ("Cannot Delete!! \n");
                       }
                }
              ptr1 -> next = ptr->next;
                free (ptr);
                printf ("\n Deleted Node : > %d", locat+1);
     }
        void display() {
                 struct node *ptr;
                  ptr =head;
                  if (ptr == NULL ) {
                      printf ("EMPTY !!! \n");
                      }
                   else
                      {
                   while(ptr!=NULL) {

                     printf("%d\n",ptr->data);
                       ptr = ptr -> next;
                   }
              }
          }
```