Lab 6 Linked List.

```
struct node
{
  int info;
  struct node *next;
  };
struct node *start = NULL;
  int main()
  {
   int choice;
   while (1) {
    printf ("\n Main ");
    printf ("\n 1. Create ");
    printf ("\n 2. Display ");
    printf ("\n 3. Insert at the beginning");
     printf ("\n 4. Insert at the end");
    printf ("\n 5. Insert at specified position");
    printf ("\n 6. Delete from beginning");
    printf ("\n 7. Delete from the end");
    printf ("\n 8. Delete from specified position");
    printf ("\n 9. Exit");
    printf ("\n Enter your choice :");
     scanf ("%d", &choice);
      switch (choice)
          {
            case 1: create();
                    break;
```

```
case 2:
        display();
        break;
case 3:  insert_begin();
        break;
case 4:  insert_end();
        break;
case 5:  insert_pos();
        break;
case 6:  delet_begin();
        break;
case 7:  delet_end();
        break;
case 8:  delck_pos();
        break;
case 9:  exit(0);
        break;
default:
        printf("\n wrong choice:");
        break;
    }
}
return 0;
}
void create()
{
    struct node *temp, *ptr;
    temp = (struct node*) malloc (size of (struct node));
    if(temp==NULL)
```

```
{
    printf("out of memory:");
    exit(0);
  }
printf("Enter the data value for the node:");
scanf("%d", &temp->info);
temp->next = NULL;
if(start == NULL)
  {
    start = temp;
  }
  else {
      ptr = start;
      while(ptr->next != NULL)
      {
       ptr = ptr->next;
      }
      ptr->next = temp;
  }
}
void display()
{
  struct node *ptr;
  if(start == NULL)
    {printf("List is empty:");
      return;
    } else
      {ptr = start;
       printf("The list elements are:");
      while(ptr != NULL){
        printf("%d\t", ptr->info);
```

```
        ptr = ptr -> next;
        }
    }
}
void insert_beg_in()
{
  struct node *temp;
    temp = (struct node *)malloc(sizeof(struct node));
    if (temp == NULL)
      { printf("out of memory space:");
        return;
      }
    printf("Enter the data value for the node:");
    scanf("%d", &temp->info);
    temp->next = NULL;
    if (start == NULL)
      {   start = temp;
      } else {
          temp->next = start;
          start = temp;
          }
      }
void insert_end()
  {  struct node *temp, *ptr;
     temp = (struct node *)malloc(sizeof(struct node));
   if (temp == NULL)
    { printf("out of memory");
      return;
      }
```

```
printf(" Enter the data value for the node :");
scanf("%d", &temp->info);
temp->next = NULL;
if (start == NULL) {
    start = temp;
} else {
    ptr = start;
    while (ptr->next != NULL) {
        ptr = ptr->next;
    }
    ptr->next = temp;
}
}
void insert_pos()
{ struct node *ptr, *temp;
  int i, pos;
  temp = (struct node *) malloc (sizeof (struct node));
  if (temp == NULL)
  { printf("out of memory");
    return;
  } printf(" Enter the pos ");
  scanf("%d", &pos);
  printf(" Enter the value");
  scanf("%d", &temp->info);
  temp->next = NULL;
  if (pos == 0) {
  temp->next = start;
    start = temp;
  }
```

```
else {
  for (i=0, ptr = start; i<pos-1; i++) {
    ptr = ptr->next; if (ptr == NULL) {
    printf(" pos not found");
        return;
       }
  }
  temp->next = ptr->next;
  ptr->next = temp;
 }
}
void delete_begin()
{  struct node *ptr;
   if (ptr == NULL) {
     printf(" List is Empty:");
     return;
   }
   else {
     ptr = start;
      start = start->next;
      printf("\n deleted element is: %d", ptr->info);
      free(ptr);
       }
}
   void  delete_end() {
    struct  node  *temp, *ptr;
      if (start == NULL)
      { printf(" Li is  Empty");
        exit(0);
      }
```

```
else if (start->next == NULL)
  { ptr = start;
    start = NULL;
    printf("\n deleted element is : %d ", ptr->info );
    free(ptr);
  }
  else {
    ptr = start;
    while(ptr->next != NULL)
        { temp = ptr;
          ptr = ptr->next;
        }
    temp->next = NULL;
    printf("\n The deleted element is : %d ", ptr->info);
    free(ptr);
  }
}
void delete_pos()
{ int i, pos;
  struct node *temp, *ptr;
  if (start == NULL)
  { printf("\n List is empty");
    exit(0);
  } else {
    printf("\n Enter the position : ");
    scanf("%d", &pos);
    if(pos == 0) {
    ptr = start;
    start = start->next;
    printf("\n deleted element: %d", ptr->info);
}
```

```
else {
    ptr = start;
for(i=0; i<pos; i++){
    temp = ptr; ptr = ptr->next;
        if(ptr == NULL)
        {
        printf("pos not found:");
            return;
        }
    }
    temp->next = ptr->next;
    printf("deleted element :%d", ptr->no);
    free(ptr);
    }
}
```