NAME: Ramya Ramesh
USN: 1BM19CH038

Lab 6 Singly Linked List Implementation & Insertion of Node at Beginning, End and at any specified position; Deletion of Node from Beginning, End and at any position.

Pseudocode

```
/*Node Implementation*/
struct node {
        int info ;
        struct node *next ;
    } ;
struct node * start = NULL;

/* Create*/
    struct node *temp, *ptr ;
    temp =(struct node *)malloc (sizeof (struct node));
    temp → next = NULL
    if (start == NULL)
    {
        start = temp ;
    }
    else {
        ptr = start ;
        while (ptr → next != NULL)
        {
            ptr = ptr → next;
        }
    ptr → next = temp ;
}
```

/* Display */

```
struct node * ptr;
if (start == NULL) {
    printf("\n list is empty!");
    return;
}
else {
    ptr = start;
    printf("\n List elements -> ");
    while (ptr != NULL)
    {
        printf("%d", ptr -> info);
        ptr = ptr -> next;
    }
}
```

/* Insert node at beginning */

```
{
    struct node * temp;
    temp = (struct node *) malloc (sizeof(struct node));
    printf("\n Enter value of node : ");
    scanf("%d", &temp -> info);
    temp -> next = NULL;
    if (start == NULL) {
        start = temp;
    }
    else
    {
        temp -> next = start;
        start = temp;
    }
```

```
/* Insert node at the end */
struct node *temp = (struct node*) malloc (sizeof (struct node)), *ptr;
printf ("\n Enter the value for 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;
}

/* Insert at any specified position */
struct node * ptr, * temp;
int i, pos;
temp = (struct node *) malloc (sizeof (struct node));
printf("\n Enter position for new node to be
    inserted : ");
scanf("%d", &pos);
printf("\n Enter the value of node : ");
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;
        }
        temp->next = ptr->next;
        ptr->next = temp;
    }

/* Deletion of node from beginning */
    struct node *ptr;
    if (start == NULL)
    {
        printf("\n List is Empty! ");
        return;
    }
    else {
        ptr = start;
        start = start->next;
        printf("\n The deleted element is : %d",
                ptr->info);
        free(ptr);
    }

/* Delete node from ending */
    struct node *temp, *ptr;
    if (start == NULL)
    {
        printf("\n List is Empty!\n");
        exit(0);
    }
```

```
else if (start->next == NULL)
    {
        ptr = start;
        start = NULL;
        printf("\n The 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);
    }

/* Delete node from any specified position */

    int i, pos;
    struct node *temp, *ptr;
    if(start == NULL)
        {
            printf("\n The List is Empty!");
            exit(0);
        }
```

```
else
  {
  printf("\n Enter position of node to be deleted: ");
  scanf("%d", &pos);
  if(pos == 0)
    {
      ptr = start;
      start = start -> next;
      printf("\n The deleted element is: %d",
      ptr -> info);
      free(ptr);
    }
  else
    {
      ptr = start;
    for(i = 0; i<pos; i++)
      {
       temp = ptr;
       ptr = ptr -> next;
       if(ptr == NULL)
        {
          printf("\n Position not found!");
          return;
        }
      temp -> next = ptr -> next;
      printf("\n The deleted element is: %d",
            ptr -> info);
        free(ptr);
    }
}
```