Single linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *next;
};
struct node *head = NULL;
struct node *tail = NULL;
void insertAtStart(int data)
{
          struct node *ptr;
          ptr = (struct node *) malloc(sizeof (struct node));
        ptr->data = data;
        if (head == NULL)
          {
                head = ptr;
                tail = ptr;
                ptr->next=NULL;
                return;
        }
        ptr->next = head;
```

```
       head = ptr;
}
void insertAtEnd(int data)
{
        struct node *ptr = (struct node *) malloc(sizeof (struct node));
       ptr->data = data;
       ptr->next = NULL;
      if (tail == NULL)
        {
             head = ptr;
             tail = ptr;
       }
      tail->next = ptr;
       tail = ptr;
}
void insertAtPos(int pos, int data)
{
        struct node *temp;
        struct node *ptr = (struct node *) malloc(sizeof (struct node));
        int i=1;
       ptr->data = data;
       ptr->next = NULL;;
  if (head == NULL || pos == 1)
        {
            if(!head)
        {
           head = ptr;
             tail = ptr;
                 return;
        }
        ptr->next = head;
        head = ptr;
        return;
}
temp = head;
while (temp)
  {
    if (pos == i + 1)
            {
                ptr->next = temp->next;
                temp->next = ptr;
                if (ptr->next == NULL)
                        tail = ptr;
                break;
        }
        i++;
        temp = temp->next;
}
```

```
        }
void deleteNode(int data)
{
        struct node *ptr, *temp;
        int res = 0;
          ptr = head;
        if (ptr->data == data)
          {
                if (ptr->next == NULL)
                    {
                        free(ptr);
                        head = tail = NULL;
                }
                head =  ptr->next;
                free(ptr);
                return;
        }
while (ptr != NULL && ptr->next != NULL)
          {
                if (ptr->next->data == data)
                    {
                        temp = ptr->next;
                        ptr->next = temp->next;
                        if (ptr->next == NULL)
                    tail = ptr;
                free (temp);
                res = 1;
            }
            ptr = ptr->next;
        }
        if (!res)
            printf("Operation failed - Give data unavailable in list\n");
}
void deleteList()
{
        struct node *ptr;
        ptr = head;
        while (ptr)
         {
            head = ptr->next;
            free(ptr);
            ptr = head;
        }
}
void display()
{
        struct node *ptr;
        ptr = head;
        while (ptr)
         {
              printf("%d ",ptr->data);
              ptr = ptr->next;
                    }
}
void isListExist()
{
        if (head)
              printf("List is available\n");
        else
              printf("List is unavailable\n");
}
int main()
{
         int flag = 1, ch, data, pos, result;
        while (flag)
         {
              printf("1. Insertion at the start of List\n");
              printf("2. Insert at the end of list\n");
              printf("3. Insert at node at the given position\n");
              printf("4. Delete node\n");
              printf("5. Delete list\n");
              printf("6. display\n");
              printf("7. Is list exists\n");
              printf("8. Exit\n");
              printf("Enter ur choice:");
              scanf("%d", &ch);
              switch (ch)
                 {
                    case 1:
                          printf("Enter data to insert into list\n");
                          scanf("%d", &data);
                          insertAtStart(data);
                          break;
                    case 2:
                          printf("Enter data to insert into list\n");
                          scanf("%d", &data);
                          insertAtEnd(data);
                          break;
                    case 3:
                          printf("Enter value for position and data\n");
                          scanf("%d%d", &pos, &data);
                          insertAtPos(pos, data);
                          break;
                    case 4:
                          printf("Enter value to delete node\n");
                          scanf("%d", &data);
```

```
                    deleteNode(data);
                    break;
                case 5:
                    deleteList();
                    break;
                case 6:
                    display();
                    break;
                case 7:
                    isListExist();
                    break;
                case 8:
                    exit(0);
                    break;
                default:
                    printf("Pleae retry once again\n");
                    break;
            }
            printf("\n\n");
        }
return 0;
}
```