

# DATA STRUCTURE LAB EXAM

University Register Number: AJC20MCA-2022

NAME : ANTO JOSEPH

CLASS : MCA

BATCH : A

ROLL-NO : 22

Date:01-07-2021

# Implement a singly linked list and remove all duplicate elements from the list

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

void binsert ();
void linsert ();
void bdelete();
void ldelete();
void removeDuplicates();
void display();

void main ()
{
    int ch =0;
    while(ch != 9)
    {

        printf("\nChoose one option from the following list ...\n");
        printf("\n===== \n");
        printf("\n1.insert begining\n2.insert at last\n3.delete from Beginning\n4.Delete from
last\n5.remove duplicate\n6.Show\n7.Exit\n");
        printf("\nenter your choice?\n");
        scanf("\n%d",&ch);
        switch(ch)
        {
            case 1:
                binsert();
                break;
            case 2:
                linsert();
                break;
            case 3:
                bdelete();
                break;
            case 4:
                ldelete();
```

```

        break;
    case 5:
        removeDuplicates() ;
        break;
    case 6:
        display();

        break;
    case 7:
        exit(0);
        break;
    default:
        printf("Please enter valid choice..");
    }
}
}

void removeDuplicates()
{
    struct node* current = head;

    struct node* next_next;

    if (current == NULL)
    {
        printf("empty");
    }

    while (current->next != NULL)
    {
        if (current->data == current->next->data)
        {
            next_next = current->next->next;
            free(current->next);
            current->next = next_next;
        }
        else
        {
            current = current->next;
        }
    }
}

void binsert()

```

```

{
    struct node *ptr;
    int item;
    ptr = (struct node *) malloc(sizeof(struct node *));
    if(ptr == NULL)
    {
        printf("overflow");
    }
    else
    {
        printf("\nEnter value\n");
        scanf("%d",&item);
        ptr->data = item;
        ptr->next = head;
        head = ptr;
        printf("\nNode inserted");
    }
}

```

```

}
void linsert()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter value?\n");
        scanf("%d",&item);
        ptr->data = item;
        if(head == NULL)
        {
            ptr -> next = NULL;
            head = ptr;
            printf("\nNode inserted");
        }
        else
        {
            temp = head;
            while (temp -> next != NULL)
            {
                temp = temp -> next;
            }
            temp->next = ptr;
            ptr->next = NULL;
            printf("\nNode inserted");
        }
    }
}

```

```

    }
}

void bdelete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nList is empty\n");
    }
    else
    {
        ptr = head;
        head = ptr->next;
        free(ptr);
        printf("\nNode deleted from the begining ...\n");
    }
}

void ldelete()
{
    struct node *ptr,*ptr1;
    if(head == NULL)
    {
        printf("\nlist is empty");
    }
    else if(head -> next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nOnly node of the list deleted ...\n");
    }

    else
    {
        ptr = head;
        while(ptr->next != NULL)
        {
            ptr1 = ptr;
            ptr = ptr ->next;
        }
        ptr1->next = NULL;
        free(ptr);
        printf("\nDeleted Node from the last ...\n");
    }
}

void display()
{

```

```
struct node *ptr;
ptr = head;
if(ptr == NULL)
{
    printf("Nothing to print");
}
else
{
    printf("\nprinting values . . . .\n");
    while (ptr!=NULL)
    {
        printf("\n%d",ptr->data);
        ptr = ptr -> next;
    }
}
}
```