

```
/*NAME-YASH GANDHI SE IT 14*/
```

```
/*Problem Statement:
```

```
Create a sorted doubly Linked List. Program will have 3 Functions
```

```
1- Create()
```

```
user will enter the No's in any order, sorted list will get created as an output.This function expected to have all three cases of insertion into sorted List
```

```
2-Reverse()
```

```
This function will simply output the list in reverse order of creation
```

```
3- Merge()
```

```
This function will merge 2 created list in sorted manner only. If there are duplicated in 2 lists, it has to be removed at the time of merging
```

```
*/
```

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
typedef struct NODE //node structure
```

```
{  
    int data;  
    struct NODE* next;  
    struct NODE* prev;  
}node;
```

```
node* add(node* head) //function to insert
```

```
{  
    int data;  
    node* newnode=(node*)malloc(sizeof(node));  
    newnode->next=NULL;  
    newnode->prev=NULL;  
    printf("\nPlease enter the data\n");  
    scanf("%d",&newnode->data);  
    data=newnode->data;
```

```
if(head==NULL) //conditions for head and sorting
```

```
{  
    head=newnode;  
}
```

```
else
```

```
{  
    node *curr=head;  
    while(curr->next!=NULL)  
    {  
        if(data<=(curr->data))
```

```

        {
            break;
        }
        curr=curr->next;
    }
    if(data==curr->data)
    {
        printf("\nData already present enter new data\n");
        return head;
    }

    if(curr==head&&data<head->data)//insert at beginning
    {
        newnode->next=head;
        head->prev=newnode;
        head=newnode;
    }
    else if(data<(curr->data))
    {
        newnode->prev=curr->prev;
        newnode->next=curr;
        curr->prev->next=newnode;
        curr->prev=newnode;
    }

    else //insert at end
    {
        curr->next=newnode;
        newnode->prev=curr;
    }
}
return head;
}

```

```

node* display(node* head) //function to display list
{
    printf("\noutput is\n");
    node *curr;
    curr=head;
    while(curr->next!=NULL)
    {
        printf("%d\n",curr->data);
        curr=curr->next;
    }
    printf("%d\n",curr->data);

    return head;
}

```

```

node* reverse(node* head,node* head1) //function to reverse and display
{

```

```

node *curr=head;
    while(curr->next!=NULL)
    {
        curr=curr->next;
    }

    node *newnode;
    newnode=(node*)malloc(sizeof(node));
    head1=newnode;
    newnode->data=curr->data;
    newnode->prev=NULL;
    newnode->next=NULL;

    while(curr->prev!=NULL)//loop for storing reverse list
    {
        newnode=(node*)malloc(sizeof(node));
        curr=curr->prev;
        newnode->data=curr->data;
        node *curr1=head1;
        while(curr1->next!=NULL)
        {
            curr1=curr1->next;
        }
        newnode->next=NULL;
        newnode->prev=curr1;
        curr1->next=newnode;
    }
    node* temp=head1;
    printf("\n The reverse list is\n");
    while(temp->next!=NULL)
    {
        printf("%d\n",temp->data);
        temp=temp->next;
    }
    printf("%d\n",temp->data);

    return head1;
}

node* create(node* head,int p)          //create function used in merging
{
    int data;
    node* newnode=(node*)malloc(sizeof(node));
    newnode->next=NULL;
    newnode->prev=NULL;
    data=p;
    newnode->data=p;

    if(head==NULL)
    {
        head=newnode;
    }
}

```

```

else
{
    node *curr=head;
    while(curr->next!=NULL)
    {
        if(data<=(curr->data))//sort condition
        {
            break;
        }
        curr=curr->next;
    }
    if(data==curr->data)
    {
        return head;
    }

    if(curr==head&&data<head->data)//at beginning
    {
        newnode->next=head;
        head->prev=newnode;
        head=newnode;
    }
    else if(data<(curr->data))//at middle
    {
        newnode->prev=curr->prev;
        newnode->next=curr;
        curr->prev->next=newnode;
        curr->prev=newnode;
    }
    else//at end
    {
        curr->next=newnode;
        newnode->prev=curr;
    }
}
return head;
}

```

```

node* merge(node* head,node* bead) //merge function
{
    node* temp=head;
    node* temp1=bead;

    while(temp1->next!=NULL)
    {
        temp=create(temp,temp1->data);
        temp1=temp1->next;
    }
}

```

```

    }
    temp=create(temp,temp1->data);

return temp;

}

void main()
{
    int x;
    while(x!=1)                //option statement
    {
        node* head=NULL;        //initializing pointer
        node* head1=NULL;
        int d=0;
        while(d!=1)
        {
            head=add(head);      //function calls
            printf("\nEnter 1 to quite else 0\n");
            scanf("%d",&d);
        }
        printf("\nOutput of list 1\n");
        head=display(head);
        head1=reverse(head,head1);

        node* bead=NULL;
        node* bead1=NULL;
        int m=0;
        while(m!=1)
        {
            bead=add(bead);
            printf("\nEnter 1 to quite else 0\n");
            scanf("%d",&m);
        }
        printf("\nOutput of list 2\n");
        bead=display(bead);
        bead1=reverse(bead,bead1);

        node* mead=NULL;
        mead=merge(head,bead);
        printf("\n after merging\n");

        mead=display(mead);

        node* mead1=NULL;
        printf("\n After reversing merged list \n");
        mead1=reverse(mead,mead1);

        printf("\n Enter 0 to do again else 1\n");
        scanf("%d",&x);
    }

}

```

/\*output

```
itlab@CL-Lab-410-13: ~/Desktop
itlab@CL-Lab-410-13:~/Desktop$ gcc doublylink.c
itlab@CL-Lab-410-13:~/Desktop$ ./a.out
Please enter the data
3
Enter 1 to quite else 0
0
Please enter the data
5
Enter 1 to quite else 0
0
Please enter the data
3
Data already present enter new data
Enter 1 to quite else 0
0
Please enter the data
7
Enter 1 to quite else 0
0
Please enter the data
1
Enter 1 to quite else 0
0
Please enter the data
9
Enter 1 to quite else 0
1
Output of list 1
```

```
1
Output of list 1
output is
1
3
5
7
9
The reverse list is
9
7
5
3
1
Please enter the data
6
Enter 1 to quite else 0
0
Please enter the data
8
Enter 1 to quite else 0
0
Please enter the data
4
Enter 1 to quite else 0
0
Please enter the data
2
Enter 1 to quite else 0
0
Please enter the data
Enter 1 to quite else 0
0
Please enter the data
10
Enter 1 to quite else 0
1
Output of list 2
output is
1
2
4
6
8
10
The reverse list is
10
8
6
4
2
1
after merging
output is
1
2
3
4
5
6
7
8
```

```
6
8
10
The reverse list is
10
8
6
4
2
1
after merging
output is
1
2
3
4
5
6
7
8
9
10
After reversing merged list
The reverse list is
10
9
8
7
6
5
4
3
2
1
Enter 0 to do again else 1
1
itlab@CL-Lab-410-13:~/Desktop$ ./a.out
```

output

```
itlab@CL-Lab-410-13:~/Desktop$ gcc doublylink.c
```

```
itlab@CL-Lab-410-13:~/Desktop$ ./a.out
```

Please enter the data

3

Enter 1 to quite else 0

0

Please enter the data

5

Enter 1 to quite else 0

0

Please enter the data

3

Data already present enter new data

Enter 1 to quite else 0

0

Please enter the data



7

Enter 1 to quite else 0  
0

Please enter the data  
1

Enter 1 to quite else 0  
0

Please enter the data  
9

Enter 1 to quite else 0  
1

Output of list 1

output is

1  
3  
5  
7  
9

The reverse list is

9  
7  
5  
3  
1

Please enter the data  
6

Enter 1 to quite else 0  
0

Please enter the data  
8

Enter 1 to quite else 0  
0

Please enter the data  
4

Enter 1 to quite else 0  
0

Please enter the data  
2

Enter 1 to quite else 0

0

Please enter the data

1

Enter 1 to quite else 0

0

Please enter the data

10

Enter 1 to quite else 0

1

Output of list 2

output is

1

2

4

6

8

10

The reverse list is

10

8

6

4

2

1

after merging

output is

1

2

3

4

5

6

7

8

9

10

After reversing merged list

The reverse list is

10

9

8

7

6

5

4  
3  
2  
1

Enter 0 to do again else 1

1  
\*/