

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

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *link;
```

```
}
```

```
typedef struct node *NODE;
```

```
NODE getnode()
```

```
{
```

```
    NODE x;
```

```
    x = (NODE) malloc (Size of (struct node));
```

```
    if (x == NULL)
```

```
    {
```

```
        printf ("Memory Full !!! \n");
```

```
        exit (0);
```

```
    }
```

```
    return x;
```

```
}
```

```
void freenode (NODE x)
```

```
{ free(x) }
```

```
NODE insert (NODE first, int item)
```

```
{
```

```
    NODE temp = getnode(), cur, prev;
```

```
    temp->info = item;
```

```
    if (first == NULL)
```

```
    {
```

```
        first = getnode();
```

```
        first->info = item;
```

return first;

if (item < first->info)

temp->link = first;

return temp;

}

cur = first;

prev = NULL;

while (cur != NULL && item > cur->info)

{

prev = cur;

cur = cur->link;

}

prev->link = temp;

temp->link = cur;

return first;

}

void reverse_list (NODE first)

{

NODE cur, temp;

cur = NULL;

while (first != NULL)

{

temp = first;

first = first->link;

temp->link = cur;

cur = temp;

}

printf ("List has been reversed successfully\n");

return cur;

}

NODE concat (NODE first, NODE second)

{

NODE cur;

if (first == NULL)

return second;

if (second == NULL)

return first;

cur = first;

while (cur->link != NULL)

cur = cur->link;

cur->link = second;

return first;

}

void display (NODE first)

{

NODE temp;

if (first == NULL)

printf ("List is EMPTY!!! \n");

for (temp = first; temp != NULL; temp = temp->link)

printf ("%d \n", temp->info);

}

}

void main ()

{

int item, choice, pos, i, n;

NODE firsta = NULL, firstb = NULL;

for (;;)

{

printf ("\n 1. Insert List 1 \n 2. Insert List 2 \n

3. Reverse List 1 \n 4. Reverse List 2 \n 5. Display

List 1 \n 6. Display List 2 \n 7. Concatenate \n

```

* 8. Exit "\n");
printf("Enter choice 0: \n");
scanf("%d", &choice);
switch (choice)
{
case 1: printf("Enter the element to be inserted \n");
        scanf("%d", &item);
        switch (ch)
        firsta = insert (firsta, item);
        break;
case 2: printf("Enter the element to be inserted \n");
        scanf("%d", &item);
        firstb = insert (firstb, item);
        break;
case 3: firsta = reverse_list (firsta);
        break;
case 4: firstb = reverse_list (firstb);
        break;
case 5: printf("List 1: \n");
        display (firsta);
        break;
case 6: printf("List 2: \n");
        display (firstb);
        break;
case 7: printf("Concatenated list: \n");
        firsta = Concat (firsta, firstb);
        display (firsta);
        break;
case 8: Exit exit(0);
default: printf("Invalid input!! \n");
}
}
}

```



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 struct node
4 {
5     int info;
6     struct node *link;
7 };
8 typedef struct node *NODE;
9 NODE getnode()
10 {
11     NODE x;
12     x = (NODE)malloc(sizeof(struct node));
13     if (x == NULL)
14     {
15         printf("MEMORY FULL!!!!\n");
16         exit(0);
17     }
18     return x;
19 }
20 NODE insert_rear(NODE first, int item)
21 {
22     NODE temp, cur;
23     temp = getnode();
24     temp->info = item;
25     temp->link = NULL;
26     if (first == NULL)
27         return temp;
28     cur = first;
29     while (cur->link != NULL)
30         cur = cur->link;
31     cur->link = temp;
32     return first;
33 }
34 void display(NODE first)
35 {
36     NODE temp;
37     if (first == NULL)
38         printf("List is EMPTY!!!\n");
39     for (temp = first; temp != NULL; temp =
temp->link)
40     {
41         printf("%d\n", temp->info);
42     }
43 }
44
45 NODE concat(NODE first, NODE second)
46 {
47     NODE cur;
48     if (first == NULL)
49         return second;
50     if (second == NULL)
51         return first;
52     cur = first;

```

```

53     while (cur->link != NULL)
54         cur = cur->link;
55     cur->link = second;
56     return first;
57 }
58 int main()
59 {
60     int item, choice;
61     NODE firsta = NULL, firstb=NULL;
62     for (;;)
63     {
64         printf("\n1:INSERT_FRONT
LIST1\n2:INSERT_FRONT LIST2\n3:DISPLAY
LIST1\n4:DISPLAY LIST2\n5:CONCATENATE AND
DISPLAY\n6:EXIT\n");
65         printf("Enter choice:\n");
66         scanf("%d", &choice);
67         switch(choice)
68         {
69             case 1:
70                 printf("Enter the item\n");
71                 scanf("%d", &item);
72                 firsta = insert_rear(firsta,
item);
73                 break;
74             case 2:
75                 printf("Enter the item\n");
76                 scanf("%d", &item);
77                 firstb = insert_rear(firstb,
item);
78                 break;
79             case 3:
80                 printf("list 1:\n");
81                 display(firsta);
82                 break;
83             case 4:
84                 printf("list 2:\n");
85                 display(firstb);
86                 break;
87             case 5:
88                 printf("concatenated list : \n");
89                 firsta=concat(firsta,firstb);
90                 display(firsta);
91                 break;
92             case 6:
93                 exit(0);
94             default:printf("INVALID INPUT!!
\n");
95
96
97         }
98     }
99 }

```

```
1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
```

Enter choice:

1

Enter the item

10

```
1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
```

Enter choice:

1

Enter the item

20

```
1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
```

Enter choice:

2

Enter the item

30

```
1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
```

```
6:EXIT
Enter choice:
2
Enter the item
40

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
1
Enter the item
50

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
3
list 1:
10
20
50

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
4
list 2:
30
40

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
```


Enter choice:

5

concatenated list :

10

20

50

30

40

1:INSERT_FRONT LIST1

2:INSERT_FRONT LIST2

3:DISPLAY LIST1

4:DISPLAY LIST2

5:CONCATENATE AND DISPLAY

6:EXIT

Enter choice:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 struct node{
4     int info;
5     struct node *link;
6 };
7 typedef struct node *NODE;
8
9
10 NODE getnode()
11 {
12     NODE x;
13     x=(NODE)malloc(sizeof(NODE));//
14     if(x==NULL)
15     {
16         printf("memory full \n");
17         exit(0);
18     }
19     return x;
20 }
21
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26
27
28 NODE insert(NODE first,int item)
29 {
30     NODE temp=getnode(),cur,prev;
31     temp->info=item;
32     if(first==NULL)
33     {
34         first=getnode();
35         first->info=item;
36         return first;
37     }
38     if(item<first->info)
39     {
40         temp->link=first;
41         return temp;
42     }
43     cur=first;
44     prev=NULL;
45     while(cur!=NULL&&item>cur->info)
46     {
47         prev=cur;
48         cur=cur->link;
49     }
50     prev->link=temp;
51     temp->link=cur;
52     return first;
53 }

```

```

54 NODE reverse_list(NODE first)
55 {
56     NODE cur,temp;
57     cur = NULL;
58     while(first!=NULL)
59     {
60         temp = first;
61         first=first->link;
62         temp->link=cur;
63         cur=temp;
64     }
65     printf("List has been reversed
successfully\n");
66     return cur;
67 }
68 void display(NODE first)
69 {
70     if(first==NULL)
71     {
72         printf("List is empty\n");
73         return;
74     }
75     printf("Elements of the list are : \n");
76     for(NODE i=first;i!=NULL;i=i->link)
77         printf("%d\n",i->info);
78 }
79 int main()
80 {
81     int item,ch;
82     NODE first=NULL;
83     for(;;)
84     {
85         printf("\n1.Insert and
Sort\n2.Reverse\n3.Display\n");
86         scanf("%d",&ch);
87         switch(ch)
88         {
89             case 1:
90                 printf("Enter element to be
inserted\n");
91                 scanf("%d",&item);
92                 first = insert(first,item);
93                 break;
94             case 2:
95                 first=reverse_list(first);
96                 break;
97             case 3:
98                 display(first);
99                 break;
100             default:return 0;
101         }
102     }
103 }

```


1.Insert and Sort
2.Reverse
3.Display

1

Enter element to be inserted

10

1.Insert and Sort
2.Reverse
3.Display

1

Enter element to be inserted

20

1.Insert and Sort
2.Reverse
3.Display

1

Enter element to be inserted

9

1.Insert and Sort
2.Reverse
3.Display

3

Elements of the list are :

9

10

20

1.Insert and Sort
2.Reverse
3.Display

2

List has been reversed successfully

1.Insert and Sort
2.Reverse
3.Display

3

Elements of the list are :

20

10

9

1.Insert and Sort
2.Reverse
3.Display