

→ DS Lab Programs 7

② #include <stdio.h>

struct Node {

int info;

struct Node \*link;

};

typedef struct Node node;

node getnode() {

node n;

n=(node)malloc(sizeof(node));

if (n==NULL) {

printf("Memory full\n");

exit(0);

}

return n;

};

void freeNode (node n) {

free(n);

};

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* & *cur* *item*  $\rightarrow$  *cur*  $\rightarrow$  *uiifa*) {

*prev* = *cur*;

*cur* = *cur*  $\rightarrow$  *link*;

}

*prev*  $\rightarrow$  *link* = *temp*;

*temp*  $\rightarrow$  *link* = *cur*;

    set *cur* ~~*temp*~~; *first*;

}

node reverse - list (Node *first*) {

    node *cur*, *temp*;

*cur* = *NULL*;

    while (*first* != *NULL*) {

*temp* = *first*;

*first* = *first*  $\rightarrow$  *link*;

*temp*  $\rightarrow$  *link* = *cur*;

*cur* = *temp*;

}

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

    set *cur*;

}

void display (Node *first*) {

    if (*first* == *NULL*) {

        printf ("List is empty \n");

    set *cur*;

}

    printf ("Elements of the list: \n");

    for (node *i* = *first*; *i* != *NULL*; *i* = *i*  $\rightarrow$  *link*)

        printf (" %d \n", *i*  $\rightarrow$  *uiifa*);

3

```
printf("Enter first node value: ");
scanf("%d", &first);
printf("Enter second node value: ");
scanf("%d", &second);
node *temp = NULL;
temp->data = first;
temp->link = second;
printf("Linked list created successfully.\n");
return temp;
```

```
int main() {
    int item, choice, pos, i, n;
    node *firsta = NULL, *firstb = NULL;
    for (;;) {
        printf("1. Insert front - 2. Insert front\n"
               "- 3. Concatenate - 4. Reverse\n"
               "5. display");
        scanf("%d", &choice);
        switch (choice) {
            case 1: printf("Enter item\n");
                scanf("%d", &item);
                firsta = insert(firsta, item);
                break;
            case 2: printf("Enter item\n");
                scanf("%d", &item);
                firstb = insert(firstb, item);
                break;
            case 3: printf("concatenated list:\n");
                firsta = concat(firsta, firstb);
                display(firsta);
                break;
        }
    }
}
```

case 4: firsta = reverse list (firsta);  
firstb = reverse - list (firstb);  
break;

case 5: display (firsta);  
display (firstb);  
break;

default: return 0;

y

y

y

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node{
5     int info;
6     struct node *link;
7 };
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=temp;
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)
```

```
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
55
56 NODE reverse_list(NODE first)
57 {
58     NODE cur,temp;
59     cur = NULL;
60     while(first!=NULL)
61     {
62         temp = first;
63         first=first->link;
64         temp->link=cur;
65         cur=temp;
66     }
67     printf("List has been reversed successfully\n");
68     return cur;
69 }
70
71 void display(NODE first)
72 {
73     if(first==NULL)
74     {
75         printf("List is empty\n");
76         return;
77     }
78     printf("Elements of the list are : \n");
79     for(NODE i=first;i!=NULL;i=i->link)
80         printf("%d\n",i->info);
81 }
82
83 int main()
84 {
85     int item,ch;
86     NODE first=NULL;
87     for(;;)
```

```
83     int main()
84     {
85         int item,ch;
86         NODE first=NULL;
87         for(;;)
88         {
89             printf("\n1.Insert and Sort\n2.Reverse\n3.Display\n");
90             scanf("%d",&ch);
91             switch(ch)
92             {
93                 case 1:
94                     printf("Enter element to be inserted\n");
95                     scanf("%d",&item);
96                     first = insert(first,item);
97                     break;
98                 case 2:
99                     first=reverse_list(first);
100                     break;
101                 case 3:
102                     display(first);
103                     break;
104                 default: return 0;
105             }
106         }
107     }
108 }
```

```
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 NODE concat(NODE first, NODE second)
```

```
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, pos, i, n;
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:
```

```
    printf("Enter the item\n");
    scanf("%d", &item);
    firsta = insert_rear(firsta, item);
    break;
    case 2:
    printf("Enter the item\n");
    scanf("%d", &item);
    firstb = insert_rear(firstb, item);
    break;
    case 3:
    printf("list 1:\n");
    display(firsta);
    break;
    case 4:
    printf("list 2:\n");
    display(firstb);
    break;
    case 5:
    printf("concatenated list : \n");
    firsta=concat(firsta,firstb);
    display(firsta);
    break;
    case 6:
    exit(0);
    default:printf("INVALID INPUT!!\n");
}
}
return 0;
}
```

```
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
```

```
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
```