Run | Debug | Stop | Share | Save | { } Beautify

main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node
{
    int info;
    struct node*link;

};
typedef struct node*NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("memory full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_front(NODE first,int item)
{
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    return temp;
    temp->link=first;
    first=temp;
```

main.c

```
37  }
38  NODE delete_front(NODE first)
39  {
40      NODE temp;
41      if(first==NULL)
42      {
43          printf("list is empty cannot delete\n");
44          return first;
45      }
46      temp=first;
47      temp=temp->link;
48      printf("item deleted at front end is=%d \n",first->info);
49      free(first);
50      return temp;
51  }
52  NODE insert_rear(NODE first,int item)
53  {
54      NODE temp,cur;
55      temp=getnode();
56      temp->info=item;
57      temp->link=NULL;
58      if(first==NULL)
59      return temp;
60      cur=first;
61      while(cur->link!=NULL)
62      cur=cur->link;
63      cur->link=temp;
64      return first;
65  }
66  NODE delete_rear(NODE first)
67  {
68      NODE cur,prev;
69      if(first==NULL)
70      {
71          printf("list is empty cannot delete\n");
```

main.c

```
    if(first==NULL)
    {
        printf("list is empty cannot delete\n");
        return first;
    }
    if(first->link==NULL)
    {
        printf("item deleted is %d\n",first->info);
        free(first);
        return NULL;
    }
    prev=NULL;
    cur=first;
    while(cur->link!=NULL)
    {
        prev=cur;
        cur=cur->link;
    }
    printf("item deleted at rear end is %d",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}

NODE delete_pos(int pos,NODE first)
{
    NODE prev,cur;
    int count;
    if (first==NULL || pos<=0)
    {
        printf("Invalid position\n");
        return NULL;
    }
    if (pos==1)
    {
        cur=first;
```

main.c

```
102     if (pos==1)
103     {
104         cur=first;
105         first=first->link;
106         freenode(cur);
107         return first;
108     }
109     prev=NULL;
110     cur=first;
111     count=1;
112     while (cur!=NULL)
113     {
114         if (count==pos)
115         {
116             break;
117         }
118         prev=cur;
119         cur=cur->link;count++;
120     }
121     if (count!=pos)
122     {
123         printf("Invalid position\n");
124         return first;
125     }
126     prev->link=cur->link;
127     freenode(cur);
128     return first;
129 }
130 NODE insert_pos(int item,int pos,NODE first)
131 {
132     NODE temp,cur,prev;
133     int count;
134     temp=getnode();
135     temp->info=item;
136     temp->link=NULL;
137     if (first==NULL && pos==1)
```

main.c

```
138     {
139         return temp;
140     }
141     if (first==NULL)
142     {
143         printf("Invalid position\n");
144         return NULL;
145     }
146     if (pos==1)
147     {
148         temp->link=first;
149         return temp;
150     }
151     count=1;
152     prev=NULL;
153     cur=first;
154     while (cur!=NULL && count!=pos)
155     {
156         prev=cur;
157         cur=cur->link;
158         count++;
159     }
160     if (count==pos)
161     {
162         prev->link=temp;
163         temp->link=cur;
164         return first;
165     }
166     printf("Invalid position\n");
167     return first;
168 }
169
170 void display(NODE first)
171 {
172     NODE temp;
```

main.c

```
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    printf("list empty cannot display items\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}
void main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Delete at specified position\n 6:Insert at specified position\n 7:Display
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("-------\n");
        switch(choice)
        {
            case 1:printf("enter the item at front end\n");
            scanf("%d",&item);
            first=insert_front(first,item);
            break;
            case 2:first=delete_front(first);
            break;
            case 3:printf("enter the item at rear end\n");
            scanf("%d",&item);
            first=insert_rear(first,item);
            break;
            case 4:first=delete_rear(first);
            break;
            case 5:printf("Enter the position:\n");
                   scanf("%d",&pos);
```

main.c

```
    for(;;)
    {
        printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Delete at specified posi
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("-------\n");
        switch(choice)
        {
            case 1:printf("enter the item at front end\n");
            scanf("%d",&item);
            first=insert_front(first,item);
            break;
            case 2:first=delete_front(first);
            break;
            case 3:printf("enter the item at rear end\n");
            scanf("%d",&item);
            first=insert_rear(first,item);
            break;
            case 4:first=delete_rear(first);
            break;
            case 5:printf("Enter the position:\n");
                   scanf("%d",&pos);
                   first=delete_pos(pos,first);
                   break;
            case 6:printf("Enter the item and the position:\n");
                   scanf("%d%d",&item,&pos);
                   first=insert_pos(item,pos,first);
                   break;
            case 7:display(first);
            break;
            default:exit(0);
            break;
        }
    }
}
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
enter the choice
1
-------
enter the item at front end
23

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
enter the choice
1
-------
enter the item at front end
12

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
```

```
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
enter the choice
6
-------
Enter the item and the position:
67
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
enter the choice
7
-------
12
67
23

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
```

```
 7:Display_list
 8:EXIT
enter the choice
7
-------
12
67
23

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
enter the choice
5
-------
Enter the position:
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete at specified position
 6:Insert at specified position
 7:Display_list
 8:EXIT
enter the choice
7
-------
67
23
```

main.c

Ctrl+S

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node
{
    int info;
    struct node*link;

};
typedef struct node*NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("memory full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_front(NODE first,int item)
{
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    return temp;
    temp->link=first;
    first=temp;
    return first;
}
NODE delete_front(NODE first)
```

main.c

Ctrl+S

```
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("list is empty cannot delete\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("item deleted at front end is=%d \n",first->info);
    free(first);
    return temp;
}
NODE insert_rear(NODE first,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    return temp;
    cur=first;
    while(cur->link!=NULL)
    cur=cur->link;
    cur->link=temp;
    return first;
}
NODE delete_rear(NODE first)
{
    NODE cur,prev;
    if(first==NULL)
    {
        printf("list is empty cannot delete\n");
        return first;
    }
    if(first->link==NULL)
```

main.c

Ctrl+S

```
    }
    if(first->link==NULL)
    {
        printf("item deleted is %d\n",first->info);
        free(first);
        return NULL;
    }
    prev=NULL;
    cur=first;
    while(cur->link!=NULL)
    {
        prev=cur;
        cur=cur->link;
    }
    printf("item deleted at rear end is %d",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}
NODE order_list(int item,NODE first)
{
    NODE temp,prev,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL) return temp;
    if(item<first->info)
    {
        temp->link=first;
        return temp;
    }
    prev=NULL;
    cur=first;
    while(cur!=NULL&&item>cur->info)
    {
       prev=cur;
       cur=cur->link;
    }
```

main.c

Ctrl+S

```
109         cur=cur->link;
110     }
111     prev->link=temp;
112     temp->link=cur;
113     return first;
114 }
115 NODE reverse(NODE first)
116 {
117     NODE cur,temp;
118     cur=NULL;
119     while(first!=NULL)
120     {
121         temp=first;
122         first=first->link;
123         temp->link=cur;
124         cur=temp;
125     }
126     return cur;
127 }
128 NODE concat(NODE first,NODE second)
129 {
130     NODE cur;
131     if(first==NULL)
132     return second;
133     if(second==NULL)
134     return first;
135     cur=first;
136     while(cur->link!=NULL)
137     cur=cur->link;
138     cur->link=second;
139     return first;
140 }
141 void display(NODE first)
142 {
143     NODE temp;
144     if(first==NULL)
145     printf("list empty cannot display items\n");
146     for(temp=first;temp!=NULL;temp=temp->link)
```

main.c

Ctrl+S

```
    if(first==NULL)
    printf("list empty cannot display items\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}
void main()
{
    int item,choice,n,i;
    NODE first=NULL,a,b;
    for(;;)
    {
        printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Order_list\n 6:reverse_list\n 7:Concat_list\n 8:Display_list\n 9:EXIT\n")
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("------\n")
        switch(choice)
        {
            case 1:printf("enter the item at front end\n");
            scanf("%d",&item);
            first=insert_front(first,item);
            break;
            case 2:first=delete_front(first);
            break;
            case 3:printf("enter the item at rear end\n");
            scanf("%d",&item);
            first=insert_rear(first,item);
            break;
            case 4:first=delete_rear(first);
            break;
            case 5:printf("enter the item to be inserted in ordered_list\n");
            scanf("%d",&item);
            first=order_list(item,first);
            break;
            case 6:first=reverse(first);
            display(first);
            break;
            case 7:printf("Enter the no of nodes in 1\n");
```

main.c

Ctrl+S

```
            break;
            case 5:printf("enter the item to be inserted in ordered_list\n");
            scanf("%d",&item);
            first=order_list(item,first);
            break;
            case 6:first=reverse(first);
            display(first);
            break;
            case 7:printf("Enter the no of nodes in 1\n");
            scanf("%d",&n);
            a=NULL;
            for(i=0;i<n;i++)
            {
                printf("Enter the item\n");
                scanf("%d",&item);
                a=insert_rear(a,item);
            }
            printf("Enter the no of nodes in 2\n");
            scanf("%d",&n);
            b=NULL;
            for(i=0;i<n;i++)
            {
               printf("Enter the item\n");
               scanf("%d",&item);
               b=insert_rear(b,item);
            }
            a=concat(a,b);
            display(a);
            break;
            case 8:display(first);
            break;
            default:exit(0);
            break;
        }
    }
}
```

input

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
5
------
enter the item to be inserted in ordered_list
12

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
5
------
enter the item to be inserted in ordered_list
97

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
```

input

```
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
5
------
enter the item to be inserted in ordered_list
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
8
------
2
12
97

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
1
------
enter the item at front end
12

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
3
------
enter the item at rear end
56

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
```

```
 8:Display_list
 9:EXIT
enter the choice
3
------
enter the item at rear end
90

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
6
------
90
56
12

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Order_list
 6:reverse_list
 7:Concat_list
 8:Display_list
 9:EXIT
enter the choice
7
------
Enter the no of nodes in 1
3
Enter the item
23
Enter the item
12
Enter the item
15
Enter the no of nodes in 2
3
Enter the item
89
Enter the item
79
Enter the item
21
23
12
15
89
79
21
```