main.c

```c
#include<stdio.h>
#include<stdlib.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("\nMemory is full\n");
        exit(0);
    }
    return 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)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty. Cannot delete\n");
```

main.c

```
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
53  NODE IF(NODE second,int item)
54  {
55      NODE temp;
56      temp=getnode();
57      temp->info=item;
58      temp->link=NULL;
59      if(second==NULL)
60          return temp;
61      temp->link=second;
62      second=temp;
63      return second;
64  }
65
66  NODE IR(NODE second,int item)
67  {
68      NODE temp,cur;
69      temp=getnode();
70      temp->info=item;
71      temp->link=NULL;
72      if(second==NULL)
73          return temp;
74      cur=second;
75      while(cur->link!=NULL)
76          cur=cur->link;
77      cur->link=temp;
78      return second;
79  }
80
81  NODE reverse(NODE first)
82  {
83      NODE cur,temp;
84      cur=NULL;
```

main.c

```
83      NODE cur,temp;
84      cur=NULL;
85      while(first!=NULL)
86      {
87          temp=first;
88          first=first->link;
89          temp->link=cur;
90          cur=temp;
91      }
92      return cur;
93  }
94
95  NODE ascending(NODE first)
96  {
97      NODE prev=first;
98      NODE cur=NULL;
99    int temp;
100   if(first== NULL)
101   {
102       return 0;
103   }
104     else
105     {
106       while(prev!= NULL)
107       {
108           cur = prev->link;
109           while(cur!= NULL)
110           {
111               if(prev->info > cur->info)
112               {
113                 temp = prev->info;
114                 prev->info = cur->info;
115                 cur->info = temp;
116             }
117             cur = cur->link;
118           }
119           prev= prev->link;
120       }
121   }
122   return first;
123 }
124
125 NODE descending(NODE first)
```

main.c

```
123   }
124
125   NODE descending(NODE first)
126   {
127       NODE prev=first;
128       NODE cur=NULL;
129     int temp;
130     if(first==NULL)
131     {
132         return 0;
133     }
134     else
135     {
136         while(prev!= NULL)
137         {
138             cur = prev->link;
139             while(cur!= NULL)
140             {
141                 if(prev->info < cur->info)
142                 {
143                     temp = prev->info;
144                   prev->info = cur->info;
145                   cur->info = temp;
146                 }
147                 cur = cur->link;
148             }
149             prev= prev->link;
150         }
151       }
152       return first;
153   }
154
155   NODE concatenate(NODE first,NODE second)
156   {
157       NODE cur;
158       if(first==NULL)
159           return second;
160       if(second==NULL)
161           return first;
162       cur=first;
163       while(cur->link!=NULL)
164       {
165           cur=cur->link;
```

main.c

```
164     {
165         cur=cur->link;
166     }
167     cur->link=second;
168     return first;
169 }
170
171 void display(NODE first)
172 {
173     NODE temp;
174     if(first==NULL)
175         printf("List is empty. Cannot display items.\n");
176     printf("List contents are : ");
177     for(temp=first;temp!=NULL;temp=temp->link)
178     {
179         printf("\n%d",temp->info);
180     }
181 }
182 int main()
183 {
184     int item,choice,pos,element,option,choice2,item1,num;
185     NODE first=NULL;
186     NODE second=NULL;
187     for(;;)
188     {
189         printf("\n\nChoose an option");
190         printf("\n1:Insert_front \n2:Delete_front \n3:Reverse \n4:Sort
        \n5.Concatenate \n6:Display \n7:Exit\n");
191         printf("Enter the choice : ");
192         scanf("%d",&choice);
193         switch(choice)
194         {
195             case 1: printf("Enter the item at front-end : ");
196                     scanf("%d",&item);
197                     first=insert_front(first,item);
198                     printf("%d inserted at front-end.",first->info);
199                     break;
200             case 2: first=delete_front(first);
201                     break;
202             case 3: first=reverse(first);
203                     printf("List is reversed.");
204                     break;
205             case 4: printf("Press 1 for Ascending-sort and 2 for
```

```
205             case 4: printf("Press 1 for Ascending-sort and 2 for
                Descending-sort : ");
206                     scanf("%d",&option);
207                     if(option==1)
208                     {
209                         first=ascending(first);
210                         printf("List is sorted in ascending order.");
211                     }
212                     if(option==2)
213                     {
214                         first=descending(first);
215                         printf("List is sorted in descending order.");
216                     }
217                     break;
218             case 5: printf("Create a second list\n");
219                     printf("Enter the number of elements in the second list
                        : ");
220                     scanf("%d",&num);
221                     for(int i=1;i<=num;i++)
222                     {
223                         printf("\nPress 1 to Insert-front and 2 to
                            Insert-rear : ");
224                         scanf("%d",&choice2);
225                         if(choice2==1)
226                         {
227                             printf("Enter the item at front-end : ");
228                           scanf("%d",&item1);
229                             second=IF(second,item1);
230                         }
231                         if(choice2==2)
232                         {
233                             printf("Enter the item at rear-end : ");
234                           scanf("%d",&item1);
235                             second=IR(second,item1);
236                         }
237                     }
238                     first=concatenate(first,second);
239                     printf("\nThe two lists are concatenated.");
240                     break;
241             case 6: display(first);
242                   break;
243             default:exit(0);
```

```
                    second=IR(second,item1);
                }
            }
            first=concatenate(first,second);
            printf("\nThe two lists are concatenated.");
            break;
        case 6: display(first);
                break;
        default:exit(0);
                break;
    } }
}
```

```
> clang-7 -pthread -lm -o main main.c
> ./main


Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 13
13 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 15
15 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 3
List is reversed.
```

```
Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 4
Press 1 for Ascending-sort and 2 for Descending-sort : 1
List is sorted in ascending order.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
13
15
```

S.R. POOJA
1BM19CS135

4th prog.

```
#include<stdio.h>
#include<stdlib.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("\n Memory is full\n");
  exit(0); }
return x; }
NODE insert_front (NODE first, int item)
{ NODE temp;
 temp= get node();
 temp -> link= NULL;
 if (first==NULL)
{ return temp; }
temp -> link = first;
first = temp;
return first; }
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 IF (NODE second, int item)
{
```

S.R. POOJA
1BM19CS135

```
NODE temp;
temp= getnode();
temp -> info = item;
temp -> link= NULL;
if (second == NULL)
return temp;
temp -> link =second;
second = temp;
return second; }
NODE IR(NODE second, int item)
{ NODE temp, cur;
  temp = get node();
  temp -> info = item;
  temp -> link= NULL;
 if (second == NULL)
 return temp;
cur= second;
while( cur -> link != NULL)
cur= cur -> link}
cur -> link = temp;
return second; }
NODE reverse (NODE first)
{ NODE cur, temp;
   cur= NULL;
  while (first != NULL) {
temp = first;
 first = first -> link;
 temp -> link = cur;
 cur= temp; }
return cur; }
NODE ascending (NODE first)
{ NODE prev= first;
   NODE cur= NULL;
 int temp;
 if (first == NULL)
```

S.R. POOJA
1BM19CS135

```
{ return 0; }
else {
while (prev!=NULL) {
cur= prev->link;
while (cur!= NULL)
{ if (prev->info > cur->info)
{ temp=prev->info;
  prev->info = cur->info;
  cur->info = temp; }
cur= cur->link; }
prev= prev->link; }
} return first; }
NODE descending (NODE first)
{ NODE prev= first;
  NODE cur= NULL;
int temp;
  if (first==NULL) {
return 0; }
else {
while (prev!=NULL) {
cur = prev->link;
while (cur!=NULL) {
if (prev->info < cur->info) {
temp= prev->info;
prev->info = cur->info;
cur->info = temp; }
cur= cur->link; }
prev= prev->link; }
}
return first; }
NODE concatenate (NODE first, NODE second)
{ NODE cur;
 if (first==NULL)
 return second;
if (second==NULL)
```

S.R. POOJA
1BM19CS135

```
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. Cannot display items.\n");
printf("List contents are:");
for(temp= first ; temp!=NULL; temp= temp->link)
{
printf("\n%d", temp->info); }}
int main()
{ int item,choice, pos, element, option, choice2, item1, num;
NODE first=NULL;
NODE second= NULL;
for(;;)
{ printf("\n\n Choose an option");
printf("\n1: Insertfront \n 2: Delete_font \n 3: Reverse\n4. Sort
  \n 5. Concatenate \n 6: Display \n 7.: Exit\n");
printf(" Enter the choice:");
scanf("%d", &choice);
switch(choice) {
case 1: printf("Enter the item at font-end:");
        scan f("%d", & item);
        first= insert_font (first, item);
        printf("%d inserted at font end", first -> info);
        break;
case 2: first = delete_font (first);
        break;
case 3: first= reverse (first);
        printf("List is reversed.");
```

S.R.POOJA
1BM19CS135

```
        break;
case 4: printf (" Press 1 for ascending-sort and 2 for descending sort:");
        scanf ("%d", &option);
        if (option == 1)
        { first = ascending(first);
          printf ("List is sorted in ascending order.");
        }
        if (option == 2)
        { first = descending (first);
          printf ("List is sorted in descending order.");
        }
        break;
case 5: printf ("Create a second list\n");
        printf (" Enter the number of elements in second list:");
        scanf ("%d", &num);
        for (int i=1; i<=num; i++)
        { printf ("\n Press 1 to Insert front and 2 to Insert-rear:");
        scanf ("%d", &choice2);
        if choice2 == 1)
        { printf ("Enter the item at front-end:");
        scanf ("%d ,&item1);
        second = IF (second, item1); }
        if (choice2 == 2) {
        printf (" Enter the item at rear-end :");
        scanf ("%d", &item1);
        second= IR (second, item1); }
        }
        first = Concatenate (first, second);
        printf ("\n The two lists are concatenated :");
        break;
        case 6: display (first);
                break;
        default: exit(0);
                break; }
    }
}
```