File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c

```c
#include<stdio.h>
#include<conio.h>
#include<process.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("mem full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE cir_insert_front(NODE last,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
if(last==NULL)
last=temp;
temp->link=last->link;
last->link=temp;
return last;
}
NODE cir_insert_rear(NODE last,int item)
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c

```
36  NODE cir_insert_rear(NODE last,int item)
37  {
38  NODE temp;
39  temp=getnode();
40  temp->info=item;
41  if(last==NULL)
42  last=temp;
43  else
44  temp->link=last->link;
45  last->link=temp;
46  return temp;
47  }
48
49  NODE cir_delete_front(NODE last)
50  {
51  NODE temp,first;
52  if(last==NULL)
53  {
54  printf("list empty\n");
55  return NULL;
56  }
57  if(last->link==last)
58  {
59  printf("item deleted is %d",last->info);
60  freenode(last);
61  return NULL;
62  }
63  first=last->link;
64  last->link=first->link;
65  printf("item deleted at front end is %d\n",first->info);
66  freenode(first);
67  return last;
68  }
69  NODE cir_delete_rear(NODE last)
70  {
71  NODE prev;
```

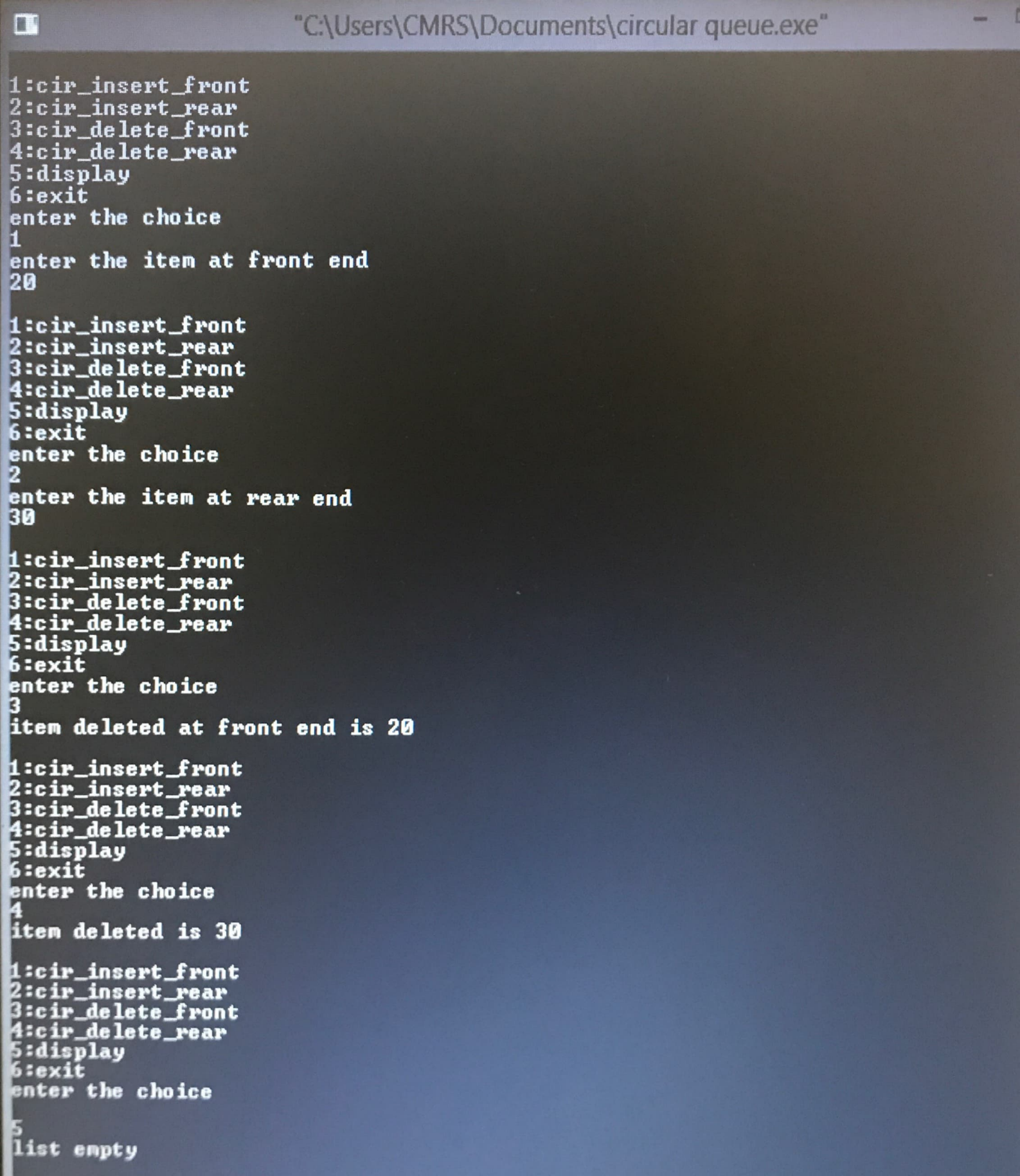
circular queue.c

```
72  if(last==NULL)
73  {
74  printf("list empty\n");
75  return NULL;
76  }
77  if(last->link==last)
78  {
79  printf("item deleted is %d\n",last->info);
80  freenode(last);
81  return NULL;
82  }
83  prev=last->link;
84  while(prev->link!=last)
85  {
86  prev=prev->link;
87  }
88  prev->link=last->link;
89  printf("item deleted at rear end is %d\n",last->info);
90  freenode(last);
91  return prev;
92  }
93  void display(NODE last)
94  {
95  NODE temp;
96  if(last==NULL)
97  {
98  printf("list empty\n");
99  return;
100
101 }
102 printf("contents of circular list are:\n");
103 for(temp=last->link;temp!=last;temp=temp->link)
104 {
105 printf("%d\n",temp->info);
106 }
107 printf("%d\n",temp->info);
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c X

```
106     }
107     printf("%d\n",temp->info);
108     }
109     void main()
110     {
111     int item,choice;
112     NODE last=NULL;
113     for(;;)
114     {
115     printf("\n1:cir_insert_front\n2:cir_insert_rear\n3:cir_delete_front\n4:cir_delete_rear\n5:display\n6:exit\n");
116     printf("enter the choice\n");
117     scanf("%d",&choice);
118     switch(choice)
119     {
120     case 1:printf("enter the item at front end\n");
121     scanf("%d",&item);
122     last=cir_insert_front(last,item);
123     break;
124
125     case 2:printf("enter the item at rear end\n");
126     scanf("%d",&item);
127     last=cir_insert_rear(last,item);
128     break;
129
130     case 3:last=cir_delete_front(last);
131     break;
132     case 4:last=cir_delete_rear(last);
133     break;
134     case 5:display(last);
135     break;
136     default:exit(0);
137     }
138     }
139     }
140
```


"C:\Users\CMRS\Documents\circular queue.exe"
1:cir_insert_front
2:cir_insert_rear
3:cir_delete_front
4:cir_delete_rear
5:display
6:exit
enter the choice
1
enter the item at front end
20
1:cir_insert_front
2:cir_insert_rear
3:cir_delete_front
4:cir_delete_rear
5:display
6:exit
enter the choice
2
enter the item at rear end
30
1:cir_insert_front
2:cir_insert_rear
3:cir_delete_front
4:cir_delete_rear
5:display
6:exit
enter the choice
3
item deleted at front end is 20
1:cir_insert_front
2:cir_insert_rear
3:cir_delete_front
4:cir_delete_rear
5:display
6:exit
enter the choice
4
item deleted is 30
1:cir_insert_front
2:cir_insert_rear
3:cir_delete_front
4:cir_delete_rear
5:display
6:exit
enter the choice
5
list empty

```c
#include<stdio.h>
#include<conio.h>
#include<process.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("mem full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
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;
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c

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

```
72      temp->link=NULL;
73      if(first==NULL&&pos==1)
74      {
75      return temp;
76      }
77      if(first==NULL)
78      {
79      printf("invalid position\n");
80      return first;
81      }
82      if(pos==1)
83      {
84      temp->link=first;
85      first=temp;
86      return temp;
87      }
88      count=1;
89      prev=NULL;
90      cur=first;
91      while(cur!=NULL&&count!=pos)
92      {
93      prev=cur;
94      cur=cur->link;
95      count++;
96      }
97      if(count==pos)
98      {
99
100     prev->link=temp;
101     temp->link=cur;
102     return first;
103     }
104     printf("invalid position\n");
105     return first;
106     }
107     NODE delete pos(int pos,NODE first)
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c X | CLLintNdel.c X

```
108 {
109 NODE cur;
110 NODE prev;
111 int count,flag=0;
112 if(first==NULL || pos<0)
113 {
114 printf("invalid position\n");
115 return NULL;
116 }
117 if(pos==1)
118 {
119 cur=first;
120 first=first->link;
121 freenode(cur);
122 return first;
123 }
124 prev=NULL;
125 cur=first;
126 count=1;
127 while(cur!=NULL)
128 {
129 if(count==pos){flag=1;break;}
130 count++;
131 prev=cur;
132 cur=cur->link;
133 }
134 if(flag==0)
135 {
136 printf("invalid position\n");
137 return first;
138 }
139 printf("item deleted at given position is %d\n",cur->info);
140 prev->link=cur->link;
141 freenode(cur);
142 return first;
143 }
```

```
144  void display(NODE first)
145  {
146  NODE temp;
147  if(first==NULL)
148  printf("list empty cannot display items\n");
149  for(temp=first;temp!=NULL;temp=temp->link)
150
151  {
152  printf("%d",temp->info);
153  }
154  }
155  void main()
156  {
157  int item,choice,key,pos;
158  int count=0;
159  NODE first=NULL;
160  for(;;)
161  {
162  printf("\n 1:Insert_rear\n 2:Delete_rear\n");
163  printf("3:insert_info_position\n 4:Delete_info_position\n 5:Display_list\n 6:Exit");
164  printf("enter the choice\n");
165  scanf("%d",&choice);
166  switch(choice)
167  {
168  case 1:printf("enter the item at rear-end\n");
169  scanf("%d",&item);
170  first=insert_rear(first,item);
171  break;
172  case 2:first=delete_rear(first);
173  break;
174  case 3:printf("enter the item to be inserted at given position\n");
175  scanf("%d",&item);
176  printf("enter the position\n");
177  scanf("%d",&pos);
178  first=insert_pos(item,pos,first);
179  break;
```

```
157     int item,choice,key,pos;
158     int count=0;
159     NODE first=NULL;
160     for(;;)
161     {
162     printf("\n 1:Insert_rear\n 2:Delete_rear\n");
163     printf("3:insert_info_position\n 4:Delete_info_position\n 5:Display_list\n 6:Exit");
164     printf("enter the choice\n");
165     scanf("%d",&choice);
166     switch(choice)
167     {
168     case 1:printf("enter the item at rear-end\n");
169     scanf("%d",&item);
170     first=insert_rear(first,item);
171     break;
172     case 2:first=delete_rear(first);
173     break;
174     case 3:printf("enter the item to be inserted at given position\n");
175     scanf("%d",&item);
176     printf("enter the position\n");
177     scanf("%d",&pos);
178     first=insert_pos(item,pos,first);
179     break;
180     case 4:printf("enter the position\n");
181     scanf("%d",&pos);
182     first=delete_pos(pos,first);
183     break;
184     case 5:display(first);
185     break;
186     default:exit(0);
187     break;
188     }
189     }
190     }
191
```

C:\Users\CMRS\Documents\CLLintNdel.exe

```
 1:Insert_rear
 2:Delete_rear
 3:insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
 enter the choice
1
enter the item at rear-end
20

 1:Insert_rear
 2:Delete_rear
 3:insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
 enter the choice
3
enter the item to be inserted at given position
50
enter the position
2

 1:Insert_rear
 2:Delete_rear
 3:insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
 enter the choice
2
item deleted at rear-end is 50
 1:Insert_rear
 2:Delete_rear
 3:insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
 enter the choice
5
20
 1:Insert_rear
 2:Delete_rear
 3:insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
 enter the choice
4
enter the position
2
invalid position
```

```c
#include<stdio.h>
#include<conio.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("mem 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;
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c | LinkedListFunctions.c

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

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c | LinkedListFunctions.c

```
73      while(cur->link!=NULL)
74       cur=cur->link;
75      cur->link=temp;
76      return first;
77      }
78      NODE IR(NODE second,int item)
79      {
80      NODE temp,cur;
81      temp=getnode();
82      temp->info=item;
83      temp->link=NULL;
84      if(second==NULL)
85       return temp;
86      cur=second;
87      while(cur->link!=NULL)
88       cur=cur->link;
89      cur->link=temp;
90      return second;
91      }
92      NODE delete_rear(NODE first)
93      {
94      NODE cur,prev;
95      if(first==NULL)
96      {
97      printf("list is empty cannot delete\n");
98      return first;
99      }
100     if(first->link==NULL)
101     {
102     printf("item deleted is %d\n",first->info);
103     free(first);
104     return NULL;
105     }
106     prev=NULL;
107     cur=first;
108     while(cur->link!=NULL)
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c | LinkedListFunctions.c

```
109 {
110 prev=cur;
111 cur=cur->link;
112 }
113 printf("iten deleted at rear-end is %d",cur->info);
114 free(cur);
115 prev->link=NULL;
116 return first;
117 }
118 NODE insert_pos(int item,int pos,NODE first)
119 {
120 NODE temp;
121 NODE prev,cur;
122 int count;
123 temp=getnode();
124 temp->info=item;
125 temp->link=NULL;
126 if(first==NULL && pos==1)
127 return temp;
128 if(first==NULL)
129 {
130  printf("invalid pos\n");
131  return first;
132 }
133 if(pos==1)
134 {
135 temp->link=first;
136 return temp;
137 }
138 count=1;
139 prev=NULL;
140 cur=first;
141 while(cur!=NULL && count!=pos)
142 {
143  prev=cur;
144  cur=cur->link;
```

```
145     count++;
146   }
147   if(count==pos)
148   {
149   prev->link=temp;
150   temp->link=cur;
151   return first;
152   }
153   printf("Invalid Position \n");
154   return first;
155   }
156   NODE delete_pos(int pos, NODE first)
157   {
158   NODE cur;
159   NODE prev;
160   int count;
161   if(first==NULL || pos<=0)
162   {
163   printf("invalid position \n");
164   return NULL;
165   }
166   if (pos==1)
167   {
168   cur=first;
169    first=first->link;
170   freenode(cur);
171   return first;
172   }
173   prev=NULL;
174   cur=first;
175   count=1;
176   while(cur!=NULL)
177   {
178   if(count==pos)
179   break;  //if found
180   prev=cur;
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c | LinkedListFunctions.c

```
180     prev=cur;
181     cur=cur->link;
182     count++;
183     }
184     if(count!=pos)
185     {
186       printf("invalid position\n");
187        return first;
188     }
189     if(count!=pos)
190     {
191       printf("invalid position specified\n");
192       return first;
193     }
194
195     prev->link=cur->link;
196      freenode(cur);
197     return first;
198     }
199     NODE reverse(NODE first)
200       {
201           NODE cur,temp;
202           cur=NULL;
203           while(first!=NULL)
204           {
205                temp=first;
206                first=first->link;
207                temp->link=cur;
208                cur=temp;
209           }
210           return cur;
211     }
212     NODE asc(NODE first)
213     {
214           NODE prev=first;
215           NODE cur=NULL;
```

```
216                 int temp;
217
218     if(first== NULL) {
219             return 0;
220             }
221     else {
222           while(prev!= NULL) {
223
224             cur = prev->link;
225
226             while(cur!= NULL) {
227
228              if(prev->info > cur->info) {
229             temp = prev->info;
230             prev->info = cur->info;
231             cur->info = temp;
232              }
233              cur = cur->link;
234             }
235             prev= prev->link;
236                     }
237                 }
238                 return first;
239             }
240   NODE des(NODE first)
241   {
242         NODE prev=first;
243         NODE cur=NULL;
244                 int temp;
245
246   if(first==NULL) {
247         return 0;
248          }
249         else {
250           while(prev!= NULL) {
251
```

```
253
254         while(cur!= NULL) {
255
256          if(prev->info < cur->info) {
257         temp = prev->info;
258         prev->info = cur->info;
259         cur->info = temp;
260             }
261             cur = cur->link;
262             }
263             prev= prev->link;
264              }
265              }
266              return first;
267         }
268   NODE concate(NODE first,NODE second)
269   {
270       NODE cur;
271       if(first==NULL)
272       return second;
273       if(second==NULL)
274       return first;
275       cur=first;
276       while(cur->link!=NULL)
277       {
278           cur=cur->link;
279
280       }
281       cur->link=second;
282       return first;
283   }
284
285   void display(NODE first)
286   {
287    NODE temp;
288    if(first==NULL)
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c | LinkedListFunctions.c

```
290   for(temp=first;temp!=NULL;temp=temp->link)
291    {
292    printf("%d\n",temp->info);
293    }
294  }
295  void main()
296  {
297  int item,choice,pos,element,option,choice2,item1,num;
298  NODE first=NULL;
299  NODE second=NULL;
300
301  for(;;)
302  {
303  printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:random_position\n 6:reverse\n 7:so
304  printf("enter the choice\n");
305  scanf("%d",&choice);
306  switch(choice)
307   {
308    case 1:printf("enter the item at front-end\n");
309       scanf("%d",&item);
310       first=insert_front(first,item);
311       break;
312    case 2:first=delete_front(first);
313       break;
314    case 3:printf("enter the item at rear-end\n");
315       scanf("%d",&item);
316       first=insert_rear(first,item);
317       break;
318    case 4:first=delete_rear(first);
319       break;
320   case 5:
321   printf("press 1 to insert or 2 to delete at any desired position \n");
322   scanf("%d",&element);
323   if(element==1){
324           printf("enter the position to insert \n");
325           scanf("%d",&pos);
```

```
324             printf("enter the position to insert \n");
325             scanf("%d",&pos);
326             printf("enter the item to insert \n");
327             scanf("%d",&item);
328             first=insert_pos(item,pos,first);
329     }
330     if(element==2){
331             printf("enter the position to delete \n");
332             scanf("%d",&pos);
333             first=delete_pos(pos,first);
334     }
335             break;
336      case 6:
337             first=reverse(first);
338             break;
339      case 7:
340             printf("press 1 for ascending sort and 2 for descending sort:\n");
341             scanf("%d",&option);
342             if(option==1)
343             first=asc(first);
344             if(option==2)
345             first=des(first);
346             break;
347      case 8:
348             printf("create a second list\n");
349             printf("enter the number of elements in second list\n");
350
351             scanf("%d",&num);
352              for(int i=1;i<=num;i++){
353             printf("\n press 1 to insert front and 2 to insert rear \n");
354             scanf("%d",&choice2);
355
356                if(choice2==1){
357                printf("enter the item at front-end\n");
358                scanf("%d",&item1);
359               second=IF(second,item1);
```

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

circular queue.c | CLLintNdel.c | LinkedListFunctions.c

```
345             first=des(first);
346             break;
347       case 8:
348             printf("create a second list\n");
349             printf("enter the number of elements in second list\n");
350
351             scanf("%d",&num);
352              for(int i=1;i<=num;i++){
353             printf("\n press 1 to insert front and 2 to insert rear \n");
354             scanf("%d",&choice2);
355
356                 if(choice2==1){
357                 printf("enter the item at front-end\n");
358                 scanf("%d",&item1);
359                second=IF(second,item1);
360                 }
361
362                if(choice2==2){
363               printf("enter the item at rear-end\n");
364               scanf("%d",&item1);
365               second=IR(second,item1);
366               }
367             }
368
369            first=concate(first,second);
370            break;
371
372       case 9:display(first);
373           break;
374      default:exit(0);
375           break;
376    }
377  }
378  }
379
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
3
enter the item at rear-end
40

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
3
enter the item at rear-end
60

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
4
iten deleted at rear-end is 60
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
2
item deleted at front-end is=30

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
9
20
40
```