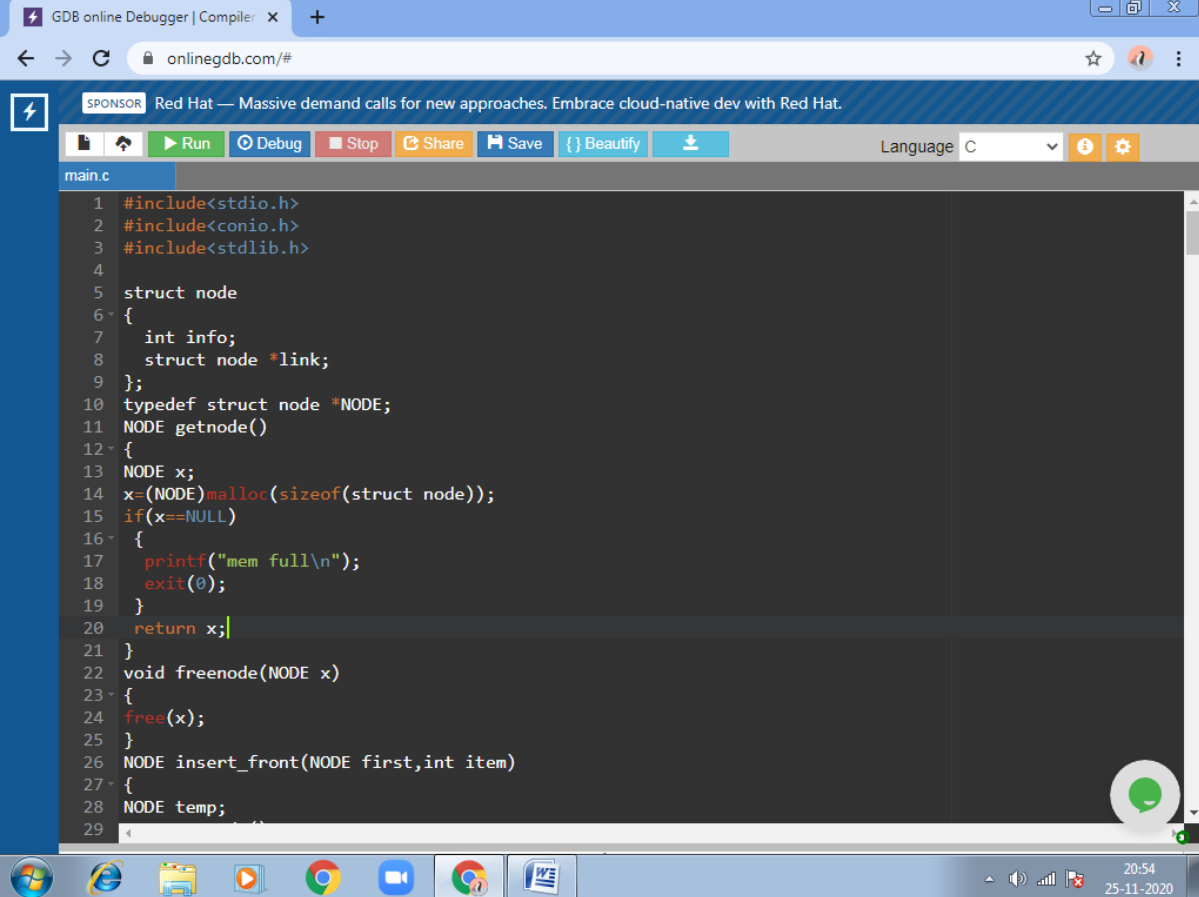


CODE:



The screenshot shows the GDB online Debugger interface in a web browser. The address bar shows `onlinegdb.com/#`. A banner for Red Hat is visible. The interface includes a toolbar with buttons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to C. The code editor displays the following C code for a linked list:

```
main.c
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4
5 struct node
6 {
7     int info;
8     struct node *link;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert_front(NODE first,int item)
27 {
28     NODE temp;
29 }
```

The Windows taskbar at the bottom shows icons for various applications and the system clock indicating 20:54 on 25-11-2020.

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
28 NODE temp;
29 temp=getnode();
30 temp->info=item;
31 temp->link=NULL;
32 if(first==NULL)
33 return temp;
34 temp->link=first;
35 first=temp;
36 return first;
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 }
```

20:54 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
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;
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 }
```

20:55 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify

Language C

main.c

```
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)
109     {
110         prev=cur;
111         cur=cur->link;
112     }
```

20:55 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify

Language C

main.c

```
111 cur=cur->link;
112 }
113 printf("item 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 }
```

20:55 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
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 }
```

20:56 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
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;
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 }
```

20:56 25-11-2020

GDB online Debugger | Compiler x +

← → ↻ onlinedb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
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 }
```

20:56 25-11-2020

GDB online Debugger | Compiler x +

← → ↻ onlinedb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
220 }
221 else {
222     while(prev!= NULL) {
223
224         cur = prev->link;
225
226         while(cur!= NULL) {
227             if(prev->info > cur->info) {
228                 temp = prev->info;
229                 prev->info = cur->info;
230                 cur->info = temp;
231             }
232             cur = cur->link;
233         }
234         prev= prev->link;
235     }
236     return first;
237 }
238 }
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     }
```

20:57 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
247     return 0;
248 }
249 else {
250     while(prev!= NULL) {
251
252         cur = prev->link;
253
254         while(cur!= NULL) {
255             if(prev->info < cur->info) {
256                 temp = prev->info;
257                 prev->info = cur->info;
258                 cur->info = temp;
259             }
260             cur = cur->link;
261         }
262         prev= prev->link;
263     }
264     return first;
265 }
266 }
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 }
```

20:57 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save {} Beautify Language C

main.c

```
274     return first;
275     cur=first;
276     while(cur->link!=NULL)
277     {
278         cur=cur->link;
279     }
280     cur->link=second;
281     return first;
282 }
283
284 void display(NODE first)
285 {
286     NODE temp;
287     if(first==NULL)
288         printf("list empty cannot display items\n");
289     for(temp=first;temp!=NULL;temp=temp->link)
290     {
291         printf("%d\n",temp->info);
292     }
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 }
```

20:57 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save Beautify Language C

main.c

```
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:rev
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);
326                 printf("enter the item to insert \n");
327                 scanf("%d",&item);
328                 first=insert_pos(item,pos,first);
329             }
```

GDB online Debugger | Compiler x +

onlinegdb.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

Run Debug Stop Share Save Beautify Language C

main.c

```
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     }
```

GDB online Debugger | Compiler x +

← → ↻ onlinedbg.com/#

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

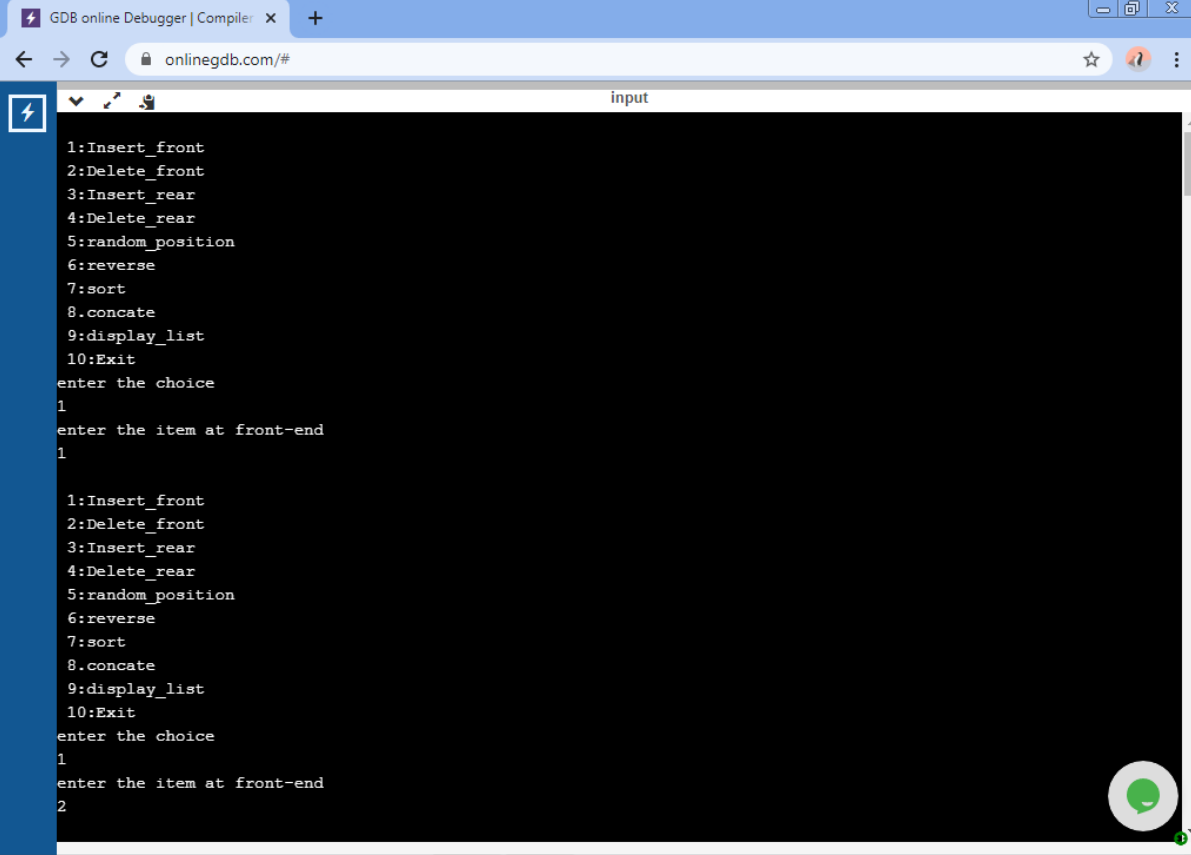
Run Debug Stop Share Save {} Beautify Language C

main.c

```
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=concat(first,second);
370 break;
371
372 case 9:display(first);
373     break;
374 default:exit(0);
375     break;
376 }
377 }
378 getch();
379 }
380
381
```

20:58 25-11-2020

OUTPUT:



The screenshot shows a web browser window titled "GDB online Debugger | Compiler" with the URL "onlinegdb.com/#". The main content area is a terminal window titled "input" with a dark background and white text. The terminal displays a menu of 10 options: 1:Insert_front, 2:Delete_front, 3:Insert_rear, 4:Delete_rear, 5:random_position, 6:reverse, 7:sort, 8.concat, 9:display_list, and 10:Exit. It then prompts "enter the choice" and receives the input "1". Next, it prompts "enter the item at front-end" and receives the input "1". The menu is repeated, followed by another "enter the choice" prompt where "1" is entered, and another "enter the item at front-end" prompt where "2" is entered. A green circular chat icon is visible in the bottom right corner of the terminal area. The Windows taskbar at the bottom shows icons for the Start menu, Internet Explorer, File Explorer, a media player, Google Chrome, a messaging app, and a document editor. The system clock in the bottom right corner indicates the time is 20:45 on 25-11-2020.

```
GDB online Debugger | Compiler x +
onlinegdb.com/#
input
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concat
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
1

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

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
enter the item at front-end
2

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

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

20:46
25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
3
enter the item at rear-end
4

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

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

20:46
25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
10:Exit
enter the choice
2
item deleted at front-end is=3

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

20:47 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
enter the choice
9
2
1

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

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

20:47 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
8

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

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
```

20:48
25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
7:sort
8:concat
9:display_list
10:Exit
enter the choice
5
press 1 to insert or 2 to delete at any desired position
1
enter the position to insert
3
enter the item to insert
6

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

20:49
25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

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

20:49 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
6
8
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8:concat
9:display_list
10:Exit
enter the choice
7
press 1 for ascending sort and 2 for descending sort:
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8:concat
9:display_list
10:Exit
enter the choice
9
1
```

20:49 25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
enter the choice
9
1
2
6
6
8

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concat
9:display_list
10:Exit
enter the choice
8
create a second list
enter the number of elements in second list
3

press 1 to insert front and 2 to insert rear
1
enter the item at front-end
1

press 1 to insert front and 2 to insert rear
1
```

20:50
25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
press 1 to insert front and 2 to insert rear
1
enter the item at front-end
3

press 1 to insert front and 2 to insert rear
2
enter the item at rear-end
5

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

20:50
25-11-2020

GDB online Debugger | Compiler x +

onlinegdb.com/#

input

```
7:sort
8.concat
9:display_list
10:Exit
enter the choice
9
1
2
6
6
8
3
1
5

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

...Program finished with exit code 0
Press ENTER to exit console.
```

20:50
25-11-2020

OBSERVATION:

```
#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;  
  
}
```

```
NODE IF(NODE second,int item)
```

```
{  
  
    NODE temp;  
  
    temp=getnode();  
  
    temp->info=item;  
  
    temp->link=NULL;  
  
    if(second==NULL)  
  
        return temp;  
  
    temp->link=second;  
  
    second=temp;  
  
    return second;  
  
}
```

```
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 IR(NODE second,int item)

```

```

{
    NODE temp,cur;
    temp=getnode();
    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 delete_rear(NODE first)
{
    NODE cur,prev;
    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 insert_pos(int item,int pos,NODE first)
{
NODE temp;

NODE prev,cur;

int count;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL && pos==1)

return temp;

if(first==NULL)
{

printf("invalid pos\n");

return first;

}

```

```

if(pos==1)
{
temp->link=first;
return temp;
}

count=1;
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid Position \n");
return first;
}

NODE delete_pos(int pos, NODE first)
{
NODE cur;
NODE prev;

```

```
int count;

if(first==NULL || pos<=0)
{
    printf("invalid position \n");
    return NULL;
}

if (pos==1)
{
    cur=first;
    first=first->link;
    freenode(cur);
    return first;
}

prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
{
    if(count==pos)
        break; //if found
    prev=cur;
    cur=cur->link;
    count++;
}

if(count!=pos)
{
    printf("invalid position\n");
```

```
    return first;
}
if(count!=pos)
{
    printf("invalid position specified\n");
    return first;
}
```

```
prev->link=cur->link;
freenode(cur);
return first;
}
```

```
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 asc(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 des(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 concate(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 empty cannot display items\n");  
    for(temp=first;temp!=NULL;temp=temp->link)  
    {  
        printf("%d\n",temp->info);  
    }  
}
```

```

void main()

{

int item,choice,pos,element,option,choice2,item1,num;

NODE first=NULL;

NODE second=NULL;


for(;;)

{

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:sort\n 8.concat\n 9:display_list\n 10:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

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("press 1 to insert or 2 to delete at any desired position \n");

```

```

scanf("%d",&element);

if(element==1){

    printf("enter the position to insert \n");

        scanf("%d",&pos);

        printf("enter the item to insert \n");

        scanf("%d",&item);

        first=insert_pos(item,pos,first);

}

if(element==2){

    printf("enter the position to delete \n");

        scanf("%d",&pos);

        first=delete_pos(pos,first);

}

        break;

case 6:

    first=reverse(first);

    break;

case 7:

    printf("press 1 for ascending sort and 2 for descending sort:\n");

    scanf("%d",&option);

    if(option==1)

        first=asc(first);

    if(option==2)

        first=des(first);

    break;

case 8:

    printf("create a second list\n");

```

```
printf("enter the number of elements in second list\n");
```

```
scanf("%d",&num);
```

```
for(int i=1;i<=num;i++){
```

```
printf("\n press 1 to insert front and 2 to insert rear \n");
```

```
scanf("%d",&choice2);
```

```
if(choice2==1){
```

```
printf("enter the item at front-end\n");
```

```
scanf("%d",&item1);
```

```
second=IF(second,item1);
```

```
}
```

```
if(choice2==2){
```

```
printf("enter the item at rear-end\n");
```

```
scanf("%d",&item1);
```

```
second=IR(second,item1);
```

```
}
```

```
}
```

```
first=concate(first,second);
```

```
break;
```

```
case 9:display(first);
```

```
break;
```

```
default:exit(0);
```

```
break;
```

```
}
```

```
}
```

```
getch();}
```