

```
#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 ("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) {
    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) {
        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, cur, prev;
    int count;

```



S.R. POOJA  
16M19C5135

```
temp = getnode();  
temp->info = item;  
temp->link = NULL;  
if (first == NULL && pos == 1) {  
    return temp; }  
if (first == NULL) {  
    printf("Invalid position\n");  
    return first; }  
if (pos == 1) {  
    temp->link = first;  
    first = temp;  
    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, flag = 0;  
    if (first == NULL || pos < 0) {  
        printf("Invalid position\n");  
        return NULL; }  
    if (pos == 1) {  
        cur = first;
```

```
temp = getnode();  
temp->info = item;  
temp->link = NULL;  
if (first == NULL && pos == 1) {  
    return temp; }  
if (first == NULL) {  
    printf("Invalid position\n");  
    return first; }  
if (pos == 1) {  
    temp->link = first;  
    first = temp;  
    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, flag = 0;  
    if (first == NULL || pos < 0) {  
        printf("Invalid position\n");  
        return NULL; }  
    if (pos == 1) {  
        cur = first;
```



```

first = first -> link;
freednode (cur);
return first; }

prev = NULL;
cur = first;
count = 1;
while (cur != NULL) {
    if (count == pos) {
        flag = 1;
        break; }
    count++;
    prev = cur;
    cur = cur -> link; }
if (flag == 0) {
    printf("Invalid position\n");
    return first; }
printf("Item deleted at given position is %d\n", cur -> info);
prev -> link = cur -> link;
freednode (cur);
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); }
}

int main() {
    int item, choice, key, pos;
    int count = 0;
    NODE first = NULL;
    for (;;) {
        printf("\n 1: Insert rear \n 2: Delete rear \n 3: Insert Front \n
        4: Delete front \n 5: Insert info position \n 6: Delete info
        position \n 7: Display list \n 8: Exit\n");

```



```
printf printf("Enter the choice:");
scanf("%d", &choice);
switch(choice) {
    case 1: printf("Enter the item at rear end\n");
            scanf("%d", &item);
            first = insert_rear(first, item);
            break;
    case 2: first = delete_rear(first);
            break;
    case 3: printf("\nEnter the item at front end\n");
            scanf("%d", &item);
            first = insert_front(first, item);
            break;
    case 4: first = delete_front(first);
            break;
    case 5: printf("Enter the item to be inserted at given position\n");
            scanf("%d", &item);
            printf("Enter the position\n");
            scanf("%d", &pos);
            first = insert_pos(item, pos, first);
            break;
    case 6: printf("Enter the position\n");
            scanf("%d", &pos);
            first = delete_pos(pos, first);
            break;
    case 7: display(first);
            break;
    default: exit(0);
            break;
}
```



```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct node{
4  int info;
5  struct node *link;
6  };
7  typedef struct node *NODE;
8  NODE getnode(){
9  NODE x;
10 x=(NODE)malloc(sizeof(struct node));
11 if(x==NULL){
12 printf("Memory full\n");
13 exit(0);
14 }
15 return x;
16 }
17 void freenode(NODE x){
18 free(x);
19 }
20 NODE insert_front(NODE first,int item){
21 NODE temp;
22 temp=getnode();
23 temp->info=item;
24 temp->link=NULL;
25 if(first==NULL)
26 return temp;
27 temp->link=first;
28 first=temp;
29 return first;
30 }
31 NODE delete_front(NODE first){
32 NODE temp;
33 if(first==NULL){
34 printf("List is empty cannot delete\n");
35 return first;
36 }
37 temp=first;
38 temp=temp->link;
39 printf("Item deleted at front end is %d\n",first->info);
40 free(first);
41 return temp;
42 }
43 NODE insert_rear(NODE first,int item){
44 NODE temp,cur;
```

```
43  NODE insert_rear(NODE first,int item){
44  NODE temp,cur;
45  temp=getnode();
46  temp->info=item;
47  temp->link=NULL;
48  if(first==NULL)
49  return temp;
50  cur=first;
51  while(cur->link!=NULL)
52  cur=cur->link;
53  cur->link=temp;
54  return first;
55  }
56  NODE delete_rear(NODE first){
57  NODE cur,prev;
58  if(first==NULL){
59  printf("List is empty cannot delete\n");
60  return first;
61  }
62  if(first->link==NULL){
63  printf("Item deleted is %d\n",first->info);
64  free(first);
65  return NULL;
66  }
67  prev=NULL;
68  cur=first;
69  while(cur->link!=NULL){
70  prev=cur;
71  cur=cur->link;
72  }
73  printf("Item deleted at rear end is %d",cur->info);
74  free(cur);
75  prev->link=NULL;
76  return first;
77  }
78  NODE insert_pos(int item,int pos,NODE first){
79  NODE temp,cur,prev;
80  int count;
81  temp=getnode();
82  temp->info=item;
83  temp->link=NULL;
84  if(first==NULL&&pos==1){
85  return temp;
86  }
```



```
85     return temp;
86 }
87 if(first==NULL){
88     printf("Invalid position\n");
89     return first;
90 }
91 if(pos==1){
92     temp->link=first;
93     first=temp;
94     return temp;
95 }
96 count=1;
97 prev=NULL;
98 cur=first;
99 while(cur!=NULL&&count!=pos){
100     prev=cur;
101     cur=cur->link;
102     count++;
103 }
104 if(
105     count==pos){
106     prev->link=temp;
107     temp->link=cur;
108     return first;
109 }
110 printf("Invalid position\n");
111 return first;
112 }
113 NODE delete_pos(int pos,NODE first){
114     NODE cur;
115     NODE prev;
116     int count,flag=0;
117     if(first==NULL || pos<0){
118         printf("Invalid position\n");
119         return NULL;
120     }
121     if(pos==1){
122         cur=first;
123         first=first->link;
124         freenode(cur);
125         return first;
126     }
127     prev=NULL;
128     cur=first;
129     count=1;
```



```
126 }
127 prev=NULL;
128 cur=first;
129 count=1;
130 while(cur!=NULL){
131     if(count==pos){
132         flag=1;
133         break;
134     }
135     count++;
136     prev=cur;
137     cur=cur->link;
138 }
139 if(flag==0){
140     printf("Invalid position\n");
141     return first;
142 }
143 printf("Item deleted at given position is %d\n",cur->info);
144 prev->link=cur->link;
145 freenode(cur);
146 return first;
147 }
148 void display(NODE first){
149     NODE temp;
150     if(first==NULL)
151         printf("List empty cannot display items\n");
152     for(temp=first;temp!=NULL;temp=temp->link){
153         printf("%d\n",temp->info);
154     }
155 }
156 int main()
157 {
158     int item,choice,key,pos;
159     int count=0;
160     NODE first=NULL;
161     for(;;){
162         printf("\n1:Insert rear\n2:Delete rear\n3:Insert front\n4:Delete front\n5:Insert info position\n6:Delete info position\n7:Display list\n8:Exit\n");
163         printf("Enter the choice: ");
164         scanf("%d",&choice);
165         switch(choice){
166             case 1:printf("Enter the item at rear end\n");
167                     scanf("%d",&item);
168                     first=insert_rear(first,item);
```



main.c

```
165  switch(choice){
166  case 1:printf("Enter the item at rear end\n");
167  scanf("%d",&item);
168  first=insert_rear(first,item);
169  break;
170  case 2:first=delete_rear(first);
171  break;
172  case 3:printf("\nEnter the item at front end\n");
173  scanf("%d",&item);
174  first=insert_front(first,item);
175  break;
176  case 4:first=delete_front(first);
177  break;
178  case 5:printf("Enter the item to be inserted at given position\n");
179  scanf("%d",&item);
180  printf("Enter the position\n");
181  scanf("%d",&pos);
182  first=insert_pos(item,pos,first);
183  break;
184  case 6:printf("Enter the position\n");
185  scanf("%d",&pos);
186  first=delete_pos(pos,first);
187  break;
188  case 7:display(first);
189  break;
190  default:exit(0);
191  break;
192  }
193  }
194  }
```



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

```
1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert info position
6:Delete info position
7:Display list
8:Exit
```

Enter the choice: 1

Enter the item at rear end

32

```
1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert info position
6:Delete info position
7:Display list
8:Exit
```

Enter the choice: 3

Enter the item at front end

12

```
1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert info position
6:Delete info position
7:Display list
8:Exit
```

Enter the choice: 2

Item deleted at rear end is 32

```
1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert info position
6:Delete info position
7:Display list
8:Exit
```



Enter the choice: 3

Enter the item at front end

12

1:Insert rear

2>Delete rear

3:Insert front

4>Delete front

5:Insert info position

6>Delete info position

7:Display list

8:Exit

Enter the choice: 2

Item deleted at rear end is 32

1:Insert rear

2>Delete rear

3:Insert front

4>Delete front

5:Insert info position

6>Delete info position

7:Display list

8:Exit

Enter the choice: 5

Enter the item to be inserted at given position

23

Enter the position

2

1:Insert rear

2>Delete rear

3:Insert front

4>Delete front

5:Insert info position

6>Delete info position

7:Display list

8:Exit

Enter the choice: 7

12

23

1:Insert rear

2>Delete rear

3:Insert front

4>Delete front

5:Insert info position

6>Delete info position

7:Display list

8:Exit

Enter the choice: