```c
#include<stdio.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("Memory is 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)
```

```
31    return temp;
32    temp->link=first;
33    first=temp;
34    return first;
35    }
36    NODE insert_rear(NODE first,int item)
37    {
38    NODE temp,cur;
39    temp=getnode();
40    temp->info=item;
41    temp->link=NULL;
42    if(first==NULL)
43    return temp;
44    cur=first;
45    while(cur->link!=NULL)
46    cur=cur->link;
47    cur->link=temp;
48    return first;
49    }
50    NODE insert_pos(int item,int pos,NODE first)
51    {
52    NODE temp,cur,prev;
53    int count;
54    temp=getnode();
55    temp->info=item;
56    temp->link=NULL;
57    if(first==NULL&&pos==1)
58    {
59    return temp;
60    }
```

```
61  if(first==NULL)
62  {
63  printf("invalid position\n");
64  return first;
65  }
66  if(pos==1)
67  {
68  temp->link=first;
69  first=temp;
70  return temp;
71  }
72  count=1;
73  prev=NULL;
74  cur=first;
75  while(cur!=NULL&&count!=pos)
76  {
77  prev=cur;
78  cur=cur->link;
79  count++;
80  }
81  if(count==pos)
82  {
83
84  prev->link=temp;
85  temp->link=cur;
86  return first;
87  }
88  printf("invalid position\n");
89  return first;
90  }
```

```
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("1.Insert_front\n2.Insert_rear\n3.Insert at given Position\n4.Display_list\n5.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:printf("enter the item at rear-end\n");
       scanf("%d",&item);
       first=insert_rear(first,item);
       break;
case 3:printf("enter the item to be inserted at given position\n");
```

```
120     case 3:printf("enter the item to be inserted at given position\n");
121            scanf("%d",&item);
122            printf("enter the position\n");
123            scanf("%d",&pos);
124            first=insert_pos(item,pos,first);
125           break;
126     case 4:display(first);
127           break;
128     default:exit(0);
129            break;
130     }
131     }
132     }
```

"C:\Users\Neha Chadaga\Desktop\LinkedListLab1.exe"

```
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
1
enter the item at front-end
1
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
1
enter the item at front-end
2
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
1
enter the item at front-end
3
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
4
3
2
1
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
2
enter the item at rear-end
4
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
```

```
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
4
3
2
1
4
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
3
enter the item to be inserted at given position
65
enter the position
3
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
4
3
2
65
1
4
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Display_list
5.Exit
enter the choice
5
```