		Page SMASH
	# include < stdio-n>	
	Struct node	
	int info;	
	Struct node * rlink -	
	tout node * Wink	
	Add to the second	
	Supodel stent nade *NODE;	
	NOOE gelrode ()	
	Alaca	
	NODE X;	,
	X = (NODE) malloc (Sige of (Struit node))	
	(C==NULI)	
	9 10 H 10	
	print ("memory full \n");	
	2	
	Date	
	neturn x;	
	1101 1000 1000 1	
	Voida fromodo (NODE X)	
	NODE dinent-front (Port item, NODE non	
	NODE dinest-front (Port item, NODE how	ud)
	9	
	NOOE temp, cur;	
	temp = getnode (): temp > info = tem = (ur = head > rlink;	
	tomp > info = tom >	
	(ui = head > rlink;	
New Year	head -> rlink = head;	
William .	temb -> rlink = cur;	
	wr -> llink = temp;	
	return head;	
	1	

Topic Control of the
NOOE dinxent rease ("int tim, NOOE hand) &
in of temp (and)
a sh = netrody ()
tanh > into = Walter,
LIA = MOOD > UINK,
acad > llink = temp;
texab -> 9 link = wad,
fomb => llink = Cur;
cur > Flink = temp;
return read;
3
NODE dedolete front (NODE hond) &
NODE Cur, next;
"of (road - reliak = = head)
40
brintle ("day empty (1")
prints ("day empty (n") =
3
cur = head > rlink;
next = are -> rlink;
hand ->rlink = rout;
next -> Wink = head;
prints ("the node deleted is old", cur-Info);
freenide (an)
Return head;
V Charles
NOOF ddelete - rear (NOOE head)
d and the state of
NOOE an brev =
"If (head -> Flink == head)
A cours - North
bout (day not a)
beinty ("day empty \n");

nown head; Cun = hoad -> llink; brow = wa -> Hick stlink = phend; points (" the node delated is look " cur - sigh beenedo (rua); notion head; NODE insent lefters (ent stem, NODE hand NOOE tomb, an, brow; if (head -> slink == head brint ("list empty sa" noturn bood; cun - hand -> Idink; while Eun 1 = hood) item = = curt - info bornk; Cun = cun -> Onlink f (cur = hand) print (" roy not found in"); notion hood; brow = (un -> Wink; print ("Enter towards left of god = "item);

```
timb = getnode ();
  rant & ( o/od", temp > into);
   brow-> Mink = temp;
   temp - llink = prov;
   cun - llink = temp;
   temb -> rlink = wa;
   notion head;
NOOE delete sperified - value (int item, NOOE hand)
 NODE posely our next
 int count;
  of Choad -> rlink == head
    brints ("list is empty");
  count = 0;
  (us = head -> hlink;
   (stile (wa! = head)
   Efitem = (un - info)
   cun=cun-relinle;
   else
    count ++;
    brev=(un -) link;
     next = cun-> rlink;
     prev-rlink-next;
     neat - Wink = pnew;
      prounde (cun);
      an = next;
```

14 (court == 0) paint (" key found "); print ("Key found at You pollings and are deleted to return head: void display (NODE head) NODE temb; if Chead - rlink = = head beint ("day empty in"); netwon point ("contents of day n"); tomb = head > nlink while (temp := head) brintle ("bod \n" temp > info): temp = temp -> rlink; void main (1) NONE hand lat: Int item choice; head = actuade (); head = wink = hand;

	Jand -> Mink - hand;
1	Jor (35)
	point ("\n 1 JF \n2. JR \n3. DF \n4. DR \n305 Sacot
	Left position (a great pention value in a Duplay a
	8- Exit (1").
	print ("Fater your chance \a"); sant ("% de", of chance);
	sant (" % d &", of choice)
	1
	cases: printy ("Enter the item at front end (a");
	sconf ("Yod" of item);
	last = dinxest front (stem, head)
	break;
	Caxe - print ("Enter the Hem at rear end \n");
	scanf ("osa", fitem);
	last = dingert (read (item, head);
	break;
	case 3 = last = ddelate front (hence);
	break;
	coul 4: last = ddelete rear (head);
	break -
	cax 5: printf ("Enter the Koy Item n");
	head = insert-left for (item, head);
	produce and entropy (ment, round)
	Care 1 bright ("C to the von store as 11)-
	cax 6: printf ("Enter the key stem n")=
	Scarf ("%d", filem)!
	head = delate - specified - value (item, head)
	Love 7 - dilda 14-11-
	(ax 7 - display (head);
	defaillult: box exit(0); 3 3 3
	adding. Total (a) , b & b

```
1 #include <stdio.h>
 2 #include <stdlib.h>
 3 struct node{
 4
   int info;
  struct node *link;
6 };
 7 typedef struct node *NODE;
 8 NODE getnode()
9 {
10
  NODE x;
11
   x=(NODE)malloc(sizeof(NODE));//
  if(x==NULL)
12
13
14
     printf("memory full \n");
15
    exit(0);
16
17
    return x;
18 }
19
20 void freenode(NODE x)
21 {
22
   free(x);
23 }
24
25 NODE insert_front(NODE first,int item)
26 {
27
  NODE temp = getnode();
28
  temp->info = item;
29 temp->link = NULL;
30
   if(first == NULL)
31
   return temp;
32
   temp->link=first;
33
    return temp;
34 }
35
36 NODE delete_front(NODE first)
37 {
38 NODE temp;
39
   if(first == NULL)
40
41
     printf("List is empty\n");
42
     return first;
43
    printf("Item deleted %d",(first->info));
44
45
   temp = first;
   temp=temp->link;
46
47
    free(first);
48
    return temp;
49 }
50
51 NODE insert_rear(NODE first,int item)
53 NODE temp = getnode(),cur;
```

```
53
     NODE temp = getnode(),cur;
 54
    temp->info=item;
 55
     temp->link = NULL;
    if(first==NULL)
 56
 57
     return temp;
 58
     cur=first;
 59
    while(cur->link!=NULL)
 60
    cur=cur->link;
 61
    cur->link=temp;
 62 return first;
63 }
 64
 65 NODE delete_rear(NODE first)
 66 {
    NODE cur=first,prev=NULL;
 67
    if(first==NULL)
 68
 69
     {
      printf("List empty\n");
 70
 71
      return NULL;
 72
 73
     if(first->link==NULL)
 74
 75
      printf("item deleted is
   %d\n",first->info);
 76
      free(first);
 77
      return NULL;
 78
    }
 79
    while(cur->link!=NULL)
 80
 81
     prev=cur;
     cur=cur->link;
 82
 83
 84
    printf("Item deleted is %d\n",(cur->info));
 85
    free(cur);
 86
     prev->link=NULL;
     return first;
 87
 88 }
 89
 90 void display(NODE first)
 91 {
 92
    if(first==NULL)
 93
 94
      printf("List is empty\n");
 95
     return;
 96
     printf("Elements of the list are : \n");
 97
 98
     for(NODE i=first;i!=NULL;i=i->link)
99
     printf("%d\n",i->info);
100 }
101 NODE insert_pos(NODE first,int item,int pos)
102 {
103
    int c=1;
104
    NODE temp = getnode(),cur,prev;
```

```
105
     temp->info=item;
106
     if(pos==1)
107
108
      temp->link=first;
109
      return temp;
110
111
     cur=first;
112
     prev=NULL;
113
     while(cur!=NULL)
114
     {
115
      if(pos==c)
      {
116
117
        prev->link=temp;
        temp->link=cur;
118
119
        return first;
120
      }
121
      C++;
122
      prev=cur;
123
      cur=cur->link;
124
     printf("Invalid position\n");
125
126
     return first;
127 }
128
129 NODE delete_pos(NODE first,int pos)
130 {
131
    NODE cur, prev;
132
    int c=1;
    if(first==NULL||pos<0)
133
134
     {
135
      printf("Invalid position\n");
136
      return NULL;
137
138
     if(pos==1)
139
140
      free(first);
141
      return NULL;
142
143
     cur=first;
144
     prev=NULL;
145
     while(cur!=NULL)
146
     {
147
      if(c==pos)
148
      {
149
        printf("Element deleted is
   %d",cur->info);
150
        prev->link=cur->link;
151
        free(cur);
152
        return first;
      }
153
154
      prev=cur;
155
      cur=cur->link;
156
      C++;
```

```
157
158 printf("Element not found\n");
159 return first;
160 }
161 NODE delete_key(NODE first,int key)
162 {
163
    NODE prev, cur;
164
     if(first==NULL)
165
166
      printf("List is empty\n");
167
      return NULL;
168
169
    if(key==first->info)
170
     cur=first;
171
172
     first=first->link;
173
      free(cur);
174
      printf("Element deleted successfully\n");
175
      return first;
176
    }
177
     cur=first;
178
     prev=NULL;
179
    while(cur!=NULL)
180
     {
181
      if(key==cur->info)
182
183
        printf("Item deleted successfully\n");
184
        prev->link=cur->link;
185
        free(cur);
186
       return first;
      }
187
188
      prev=cur;
189
     cur=cur->link;
190
191
    if(cur==NULL)
192
     printf("Element not found \n");
193
    return first;
194 }
195
196 NODE reverse list(NODE first)
197 {
198 NODE cur, temp;
199
    cur = NULL;
    while(first!=NULL)
200
201
202
     temp = first;
203
      first=first->link;
204
      temp->link=cur;
205
      cur=temp;
206
207
     printf("List has been reversed
    successfully\n");
208
    return cur;
```

```
208
     return cur;
209
210
211
212 int main()
213 {
214
     int item, ch, pos;
     NODE first=NULL;
215
216 for(;;)
217
    {
218
      printf("\n1.Insert front\n2.Delete
    front\n3.Insert rear\n4.delete
    rear\n5.Insert_pos\n6.Delete_pos\n7.Delete
    key\n8.Display\n9.Reverse\n10.Exit\n");
      scanf("%d", &ch);
219
      switch(ch)
220
221
      {
222
        case 1:
         printf("Enter element to be
223
   inserted\n");
         scanf("%d",&item);
224
225
         first = insert_front(first,item);
226
         break;
       case 2:
227
         first = delete_front(first);
228
229
         break:
230
        case 3:
         printf("Enter element to be
231
   inserted\n");
         scanf("%d",&item);
232
         first = insert rear(first,item);
233
234
         break;
        case 4:
235
236
         first = delete_rear(first);
237
         break;
        case 5:
238
         printf("Enter element to be
239
   inserted\n");
         scanf("%d",&item);
240
         printf("Enter position\n");
241
         scanf("%d", &pos);
242
         first = insert_pos(first,item,pos);
243
244
         break:
245
        case 6:
246
         printf("Enter position\n");
```

```
scanf("%d",&pos);
247
          first = delete_pos(first,pos);
248
249
          break:
250
        case 7:
251
          printf("Enter element to be
    deleted\n");
          scanf("%d",&item);
252
          first = delete_key(first,item);
253
254
          break;
255
        case 8:
          display(first);
256
          break;
257
258
        case 9:
259
          first=reverse_list(first);
          break;
260
        default:return 0;
261
262
263
264 }
```

```
    Insert front

Delete front
Insert rear
4.delete rear
5.Insert_pos
Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
Enter element to be inserted

    Insert front

2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
Enter element to be inserted
20

    Insert front

Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
Enter element to be inserted
Enter position

    Insert front

Delete front
Insert rear
4.delete rear
```

```
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
Elements of the list are :
10
20
30
40
1.Insert front
Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
Reverse
10.Exit
Item deleted 10
1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
Item deleted is 40
1. Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
```

```
Enter element to be deleted
20
Element deleted successfully

    Insert front

Delete front
Insert rear
4.delete rear
5.Insert_pos
Delete pos
7.Delete key
8.Display
Reverse
10.Exit
8
Elements of the list are :
30

    Insert front

2.Delete front
Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
```