```
#include <stdio.h>
#include <stdlib.h>

struct node {
 int item;
 struct node *link;};
typedef struct node *NODE;
NODE getnode ( ){
  NODE x;
  x =(NODE) malloc ( sizeof (struct node));
  if (x == NULL)
  { printf(" memory full");
    exit(0);
  }
 return x;
}

void freenode (NODE x){
 free (x);
}

 NODE insert front (NODE front, 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 insert_rear ( NODE first, int item){
NODE cur, temp;
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_front(NODE first) {
NODE temp;
if (first == NULL) {
printf(" list is empty");
return first; }
temp = first;
temp = temp -> link;
printf(" item deleted at front end");
free (first);
return temp;
}

NODE delete_rear ( NODE first) {
NODE cur, prev;
if (first == NULL) {
printf("list is empty");
return first; }
```

```
    if (first->link == NULL)
    { printf("item deleted is %d", first->info);
      free(first);
      return NULL;
    }
    prev = NULL;
    cur = first;
    while(cur->link != NULL)
      { prev = cur;
        cur = cur->link;
      }
    printf("item deleted is %d", cur->info);
    free(cur);
    Prev->link = NULL;
    return first;
}

void display(NODE first){
    NODE temp;
    if (first == NULL)
    printf("list is empty");
    for(temp = first; temp != NULL; temp =
        temp->link){
        printf("%d \n", temp->info);
    }
```

```
NODE insert_pos(int pos, NODE first){
NODE temp, cur, prev;
int count;
temp = getnode();
temp->info = item;
temp->link = NULL;
if(first == NULL && pos == 1
& return temp; }
if(first == NULL)
{ printf("Invalid position");
  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");
return first;
}

NODE delete_pos(NODE first, int pos){
    NODE cur;
    NODE prev;
    int count = 0, flag = 0;
    if (first == NULL || pos <0)
    { printf(" invalid pos");
    }
    if (pos == 1){
    cur = first;
    first = first -> link;
    freenode (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"); return first;}
printf (" item deleted is %d", cur -> info);
prev -> link = cur -> link;
freenode (cur);
return first;
}
```

```
P void main ( ) {
  int item, pos, ch;
  NODE first = NULL;
  for (;;) {
  printf ("Enter your choice. 1. Insert-front
  \n 2 insert rear \n3 delete front \n 4
   delete rear \n 4 display \n5 insert
  at specific position \n 6 ~~insert a~~
  delete at specific position ");
  scanf ("%d", &ch);
  switch (ch) {
  case 1 : printf ("Enter item to be inserted");
           scanf("%d", &item);
           first = insert-front (first, item);
           break;
  case 2 : printf("Enter item to be inserted");
           scanf("%d", &item);
           first = insert-rear (first, item);
           break;

case 3 :   first = delete-front ( first);
           break;
case 4 :  first = delete-rear (first);

case 5 :  display ( first);
          break;
```

```
case 6: printf("Enter item to be inserted");
        scanf("%d", &item);
        printf("Enter position");
        scanf("%d", &pos);
        first = insert_pos(first, item, pos);
        break;
case 7: printf("Enter ~~item~~ position to be deleted");
        scanf("%d", &pos);
        first = delete_pos(first, pos);
}}}
```