```
#include <stdio.h>

struct node {
    int value;
    node *next;
};

typedef struct node *NODE

NODE getNode(){
    NODE temp;
    temp = (NODE) malloc(sizeof(struct node));
    if(temp == NULL){
        printf("Memory full \n");
        return NULL;
    }
    return temp;
}
void freeNode(NODE temp){
    free(temp);
}
NODE insertFront(NODE first){
    NODE temp;
    temp = getNode();
    int num;
    scanf("%d", &num);
    temp -> info = num;
    temp -> next = NULL;
    if(first == NULL){
      return temp;
    }
```

```
    temp -> next = first;
    first = temp;
    return first;
}
NODE insertPos(NODE first, int pos){
    int num;
    scanf("%d", &num);
    NODE temp, curr, prev;
    temp = getNode();
    temp -> value = num;
    temp -> next = NULL;
    if(first == NULL){
        return temp;
    }
    int count = 1;
    if(pos == count){
        temp -> next = first;
        first = temp;
        return first;
    }
    curr = first;
    prev = NULL;
    while(curr != NULL){
        if(count == pos){
            break;
        }
        prev = curr;
        curr = curr -> next;
        count++;
    }
```

```
    if(curr == NULL){
        printf("Cannot insert\n");
        return first;
    }
    temp -> next = prev -> next;
    prev -> next = temp;
    return first;
}
NODE insertRear(NODE first){
    NODE curr, temp;
    int num;
    temp = getNode();
    scanf("%d", &num);
    temp -> value = num;
    temp -> next = NULL;
    if(first == NULL){
        return temp;
    }
    curr = first;
    while(curr -> next != NULL){
        curr = curr -> next;
    }
    curr -> next = temp;
    return first;
}
NODE deleteFront(NODE first){
    NODE temp;
    if(first == NULL){
        printf("List is empty\n");
        return first;
```

```
if(first->next==NULL){
    printf("Deleted element = %d\n", first->value);
    freeNode(first);
    return NULL;
}
temp = first;
temp = temp->next;
printf("Deleted element = %d\n", first->value);
freeNode(first);
return temp;
}

NODE deleteRear(NODE first){
    NODE curr;
    NODE prev;
    if(first==NULL){
        printf("List is empty\n");
        return first;
    }
    if(first->next==NULL){
        printf("%d\n", first->value);
        freeNode(first);
        return NULL;
    }
    curr = first;
    prev = NULL;
    while(curr->next!=NULL){
        prev = curr;
        curr = curr->next;
    }
```

```
    prev->next = NULL;
    printf("%d", curr->value);
    freeNode(curr);
    return first;
}
NODE deletePos(NODE first, int pos){
    NODE curr, prev;
    int count = 1;
    if(first == NULL){
        printf("List is empty\n");
        return first;
    }
    if(first->count == pos){
        printf("%d", first->value);
        curr = first;
        first = first->next;
        freeNode(curr);
    }
    curr = first;
    prev = NULL;
    while(curr != NULL){
        if(count == pos){
            break;
        }
        prev = curr;
        curr = curr->next; count++;
    }
    if(curr == NULL){
        printf("invalid\n"); return first;
    }
```

```
    prev -> link = curr -> link
    freeNode(curr);
    return first;
}

void display(NODE first){
    NODE curr = first;
    if(curr == NULL){
        pf("list is empty\n");
        return curr;
    }
    while(curr != NULL){
        printf("%d ", curr -> value);
        curr = curr -> next;
    }
    printf("\n");
}

int main(){
    int ch; NODE first = NULL;
    int pos;
    while(1){
        printf("enter\n 1-insertFront\n2-insertRear\n3-insertPos\n
                4-deleteRear\n5-deleteFront\n6-deletePos\n
                7-display\n 8-exit\n");
        scanf("%d", &ch);
        switch(ch){
```

```
case 1:
    first = insertFront(first);
    break;
case 2:
    first = insertRear(first);
    break;
case 3: scanf("%d", &pos);
    first = insertPos(first, pos);
    break;
case 4:
    first = deleteFront(first);
    break;
case 5:
    first = deleteRear(first);
    break;
case 6:
    scanf("%d", &pos);
    first = deletePost(first, pos);
    break;
case 7:
    display(first);
    break;
case 8:
    return 0;
}
}
}
```