

linked list

```
#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);
```

```
} // end of if {}
```

```
return x;
```

```
}
```

```
void freenode (NODE x) {
```

```
{ free(x); }
```

```
NODE insertfront (NODE first, int item) {
```

```
NODE temp;
```

```
temp = getnode();
```

```
temp->info = item;
```

```
temp->link = NULL;
```

```
if (first == NULL)
```

```
return temp;
```

```
temp->link = first;
```

Teacher's Signature : \_\_\_\_\_

```

    first = temp;
    return first;
}

```

```

NODE deletefront(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 and is %d \n", first->info);
    free(first);
    return temp;
}

```

```

NODE insertrear(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;
}

```

Teacher's Signature : \_\_\_\_\_



```
NODE delete_node(NODE first) {
```

```
    NODE cur, prev;
```

```
    if (first == NULL) {
```

```
        printf("cannot delete\n");
```

```
        return first;
```

```
    }
```

```
    if (first->link == NULL) {
```

```
        printf("Item deleted %d\n", first->info);
```

```
        free(first);
```

```
        return NULL;
```

```
    } else {
```

```
        prev = NULL;
```

```
        cur = first;
```

```
        while (cur->link != NULL) {
```

```
            prev = cur;
```

```
            cur = cur->link;
```

```
        }
```

```
        printf("Item deleted at %d is %d\n", cur->info);
```

```
        free(cur);
```

```
        prev->link = NULL;
```

```
        return first;
```

```
    }
```

```
NODE insertpos(int item, 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\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++; }
```

Teacher's Signature : \_\_\_\_\_

```
if (curr == pos) {
```

```
    prev->link = temp;
    temp->link = curr;
    return first;
}
```

```
{
    printf("Invalid pos \n");
    return first;
}
```

```
NODE delete_pos (int pos, NODE first) {
```

```
    NODE curr;
    NODE prev;
```

```
    int count, flag = 0;
```

```
    if (first == NULL || pos < 0) {
```

```
        printf("Invalid \n");
        return NULL;
    }
```

```
    count = 1;
```

```
    prev = NULL;
```

```
    curr = first;
```

```
    while (curr != NULL & curr != pos)
```

```
    {
        prev = curr;
```

```
        curr = curr->link;
```

```
        count++;
    }
```

```
}
```



Expt. No. ....

Date .....

Page No. ....

```
if (count == pos) .
```

```
{ prev → link = cur → link;
```

```
printf ("item deleted is %d \n", cur → info);
```

```
free (cur);
```

```
return first;
```

```
}
```

```
else
```

```
printf ("invalid pos \n");
```

```
return first;
```

```
}
```

```
void display (NODE first)
```

```
{ NODE temp;
```

```
if (first == NULL)
```

```
{ printf ("Empty"); }
```

```
for (temp = first; temp != NULL; temp = temp → link)
```

```
{ printf ("%d \n", temp → info);
```

```
} }
```

Teacher's Signature : .....

```
void main()
```

```
{ int item, choice, pos;
  Node *first = NULL;
```

```
for(;;)
```

```
{ printf("\n 1. Insert front 2. Insert rear\n
  3. Insert at specified location\n 4. Delete
  front\n 5. Delete rear\n 6. Delete
  at specified location\n 7. Display
  8. Exit\n");
```

```
printf("Enter choice\n");
scanf("%d", &choice);
```

```
switch(choice).
```

```
case 1: printf("Enter the item at front\n");
        scanf("%d", &item);
        first = insertFront(first, item);
        break;
```

```
case 2: printf("Enter the item at rear\n");
        scanf("%d", &item);
        first = insertRear(first, item);
        break;
```

Teacher's Signature : \_\_\_\_\_



```

case 3 : printf ("enter insertion at location")
          scanf ("%d", &item);
          printf ("position:\n");
          scanf ("%d", &pos);

          list = insert_pos(item, pos, list);
          break;

```

```

case 4 : list = delete_front(list);
          break;

```

```

case 5 : list = delete_rear(list);
          break;

```

```

case 6 : printf ("Enter pos : \n");
          scanf ("%d", &pos);

```

initializ

int pos;

```

list = delete_pos(pos, list);
break;

```

```

Case 7 : display(list);
          break;

```

```

default : exit(0);
break;

```

```

{

```

```

}

```

```

}

```

Teacher's Signature : \_\_\_\_\_