Lab. 10.

Singly linked list Deletion.

```
#include <stdio.h>
#include< conio.h>
#include< malloc.h>
#include <process.h>
 struct node
{  int info;
    struct node * link;
};
typedef struct node *NODE
```

1. getnode()
   creating node x = (NODE) malloc(sizeof(struct node));

2. Insertion of node.

3. Deleting node at front.
   NODE delete_front (NODE first)
   create a node temp
   check for if (first = NULL)
      print that list is empty
   temp = first;
   temp = temp -> link
   print deleted item = first -> info.

4. Deleting node at rear.
   NODE delete_rear (NODE first)
   create NODE cur, prev.
   check if (first = NULL)
      print list is empty.

prev = NULL
cur = first
traversing through the linked list
while (cur → link != NULL)
prev = cur
cur = cur → link
end of while loop.
print deleted item = cur → info
free (cur) - deleting last item from list.
prev → link = NULL.

5. Deleting item at a given position.
NODE delete_pos(int pos, NODE first)
create NODE cur, prev.
count = 1
if (first = NULL or pos <= 0)
deletion not possible.
prev = NULL
cur = first ~~count~~
traversing through list to find position - pos
while (cur → link != NULL)
if (count = pos) break
prev = cur
cur = cur → link
increament count
end of while loop
if (count != pos) position not found.
deleted item = cur → info
prev → link = cur → link;
~~dele~~ deleting node → free (cur).

6. Display items in list
void display (NODE first)
for loop (from first node to
node → link is NULL)
print all node → info .

7. void main function .
menu driven inside continuous for loop .
1. Insert 2. Delete at front
3. Delete at rear 4 Delete at pos
5. display 6. exit .
switch (choice)
case 1. Accept item from user
call function insert (first, item)
case 2. delete_front (first)
case 3. delete_rear (front)
case 4. Accept position from user
delete_pos (pos, first)
case 5. display (first).
case 6. exit(0.)
end of switch
end of for loop .