Insertion & deletion at specified position

```
#include <stdio.h>
#include <conio.h>
#include<process.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("mem full\n");
  exit(0);
}
 return x;
}
void freenode(NODE x)
{ free(x);
}
NODE insert_rear (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;
}

NODE delete_rear (NODE first)
{ NODE prev, cur;
  if (first==NULL)
{ pf(" list is empty");
  return first;
}
 if (first -> link == NUL)
{ pf(" item deleted is %d", first -> info);
  free (first);
  return NULL;
}
  prev = NULL;
  cur = first;
  while (cur -> link != NULL)
{ prev = cur;
  cur = cur -> link;
}
  pf (" item deleted at rear-end is %d", cur -> info);
  free(cur);
  prev -> link = NULL;
  return first;
}
```

```
NODE insert_pos(int item, int pos, NODE first)
{NODE temp;
 NODE prev, cur;
 int count;
 temp = getnode();
 temp -> info = item;
 temp -> link = NULL;
 if (first == NULL && pos == 1) // Inserting the node
{return temp;}                  for 1st time.
 if (first == NULL)
 { Pf("Invalid pos\n");
} return first;
```

```
if (pos==1)
{
  temp->link = first;
  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 posn\n");
return first;
}
```

```
NODE delete_rear (NODE first)
{
  NODE prev, cur;
  if (first==NULL)
  {
  pf(" list is empty");
  return first;
  }
  if (first->link==NULL)
  {
  pf("item deleted is %d", first->info);
  free (first);
  return NULL;
  }
  prev = NULL;
  cur=first;
  while (cur->link!=NULL)
  { prev=cur;
    cur =cur->link;
  }
  pf ("item deleted at rear-end is %d",cur->info);
  free(cur);
  prev->link=NULL;
  return first;
}

NODE delete_info (int key, NODE first)
{ NODE prev, cur;
  if (first==>NULL)
  { pf(" list is empty");
  } return NULL;
```

```
if (key == first -> info)
{cur = first;
 first = first -> link;
 freenode (cur)
 return first;
}
prev = NULL;
cur = first;
while (cur != NULL)
{ if (key == cur -> info)
 break;
 prev = cur
   cur = cur->link;
 }
  if (cur == NULL)
{ pf("Not present");
 return first;
}
 prev -> link = cur -> link
 pf(" key deleted is %d", cur -> info);
 free node(cur);
}return first;

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

```
void main(){
int item, choice, key, pos;
 int count=0;
NODE first= NULL;
for(;;)
{
printf("\n 1: Insert_rear\n 2: Delete_rear\n 3: Insert
      \n 4: Delete_pos\n 5: Display \n");
 printf("Enter choice");
 scanf("%d", &choice);
 switch(choice)
{ case 1: printf("Enter item at rear-end");
       scanf("%d", &item);
      first = insert_rear(first, item);
       break;
 case 2: first = delete_rear(first);
        break;
 case 3: printf("Enter item at specified position");
        scanf("%d", &item);
        printf("Enter position");
        scanf("%d", &pos);
        first= insert_pos(item, pos, first);
        break;
case 4: printf("Enter the position");
        scanf("%d", &pos);
        first = delete_info(pos, first);
   case 5: display(first);
           break;
   default: exit(0); break; }} getchar(); }
```