WEEK 10: Program to implement delete node, at ~~insertion~~ rear, front & given position in singly linked list.

Algorithm:

[Insertion of nodes as in previous program].

For deletion:

Step 1: [Method to delete at front]

```
NODE delete_front()
      {
        NODE temp;
        if(first == NULL)
          print("List is empty");
        temp=first
        ~~first~~ temp=temp->link;
            print("Item deleted at front end = ", first->info);
          free(first);
        return temp;
      }
```

Step 2: [Method to delete at rear]

```
  NODE delete_rear()
       {
         NODE cur, prev;
          if(first==NULL)
            print("List is empty");
```

```
if(first->link==NULL)
{
   print("Item deleted =", first->info);
   free(first); }
  prev=NULL;
   curr=first;
   while(curr->link!=NULL)
     {
        prev=curr;
         curr=curr->link;
     }
 print(" Item deleted at rear end =", curr->info);
 free(curr);
 prev->link=NULL;
 return first;
}
```

Step 3: [Method to delete at given position:]

```
  NODE delete_pos()
  {
     NODE curr,prev;
     int count,flag=0;
      if( first==NULL|| pos<0)
         print("Invalid position");
     if(pos==1){
         curr=first;
          first=first->link;
          freenode(curr); }
```

```
prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
   {
     if(count==pos)
     { flag=1;
       break;
     }
    count++;
    prev=cur;
    cur=cur->link;
   }
 if(flag==0)
    print("Invalid position");
 print("Item deleted at position %d is %d", pos,
                                        cur->info);
 prev->link = cur->link;
    freenode(cur);
    return first;
  }
```

Step4: [Main method]

```
   switch(choice)
  case 4: first = deletefront( first );
          break;
  Case 5: first = delete_rear( first );
          break;
```

```
case 6 : print("Enter pos : ")
         scan(pos);
         first = delete_pos(pos, first);
         break;
```