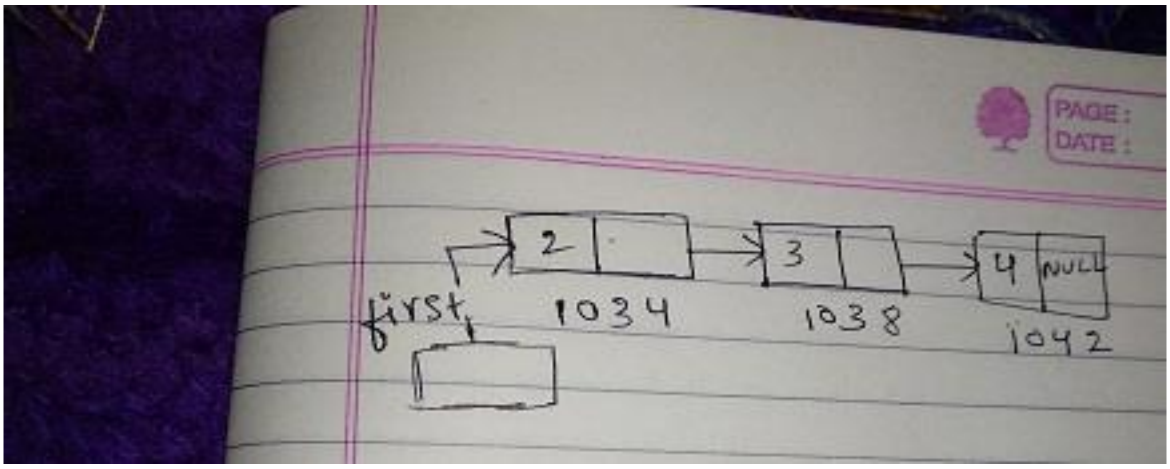
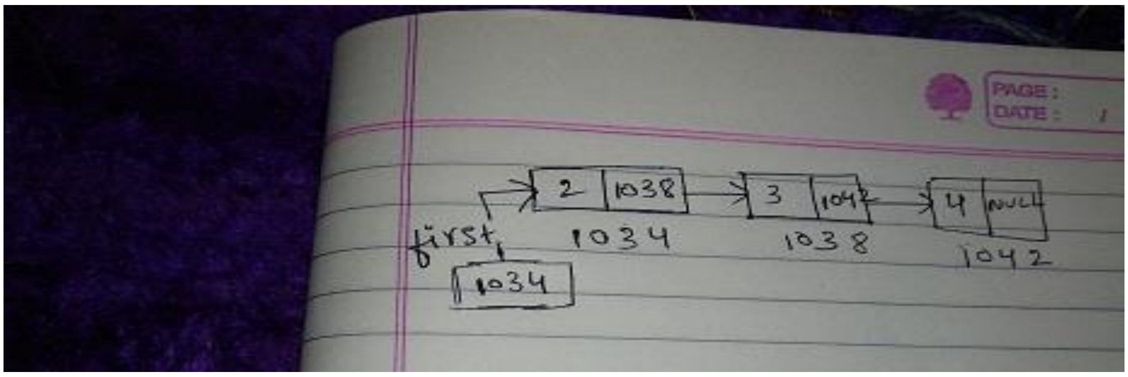
Linked list: collection of zero or more nodes where each node is connected to one or more nodes. Each node has 2 fields i,e info field and link field.

Pictorial representation:





Types of linked list:

1. Singly linked list
2. Doubly linked list
3. Circular singly linked list
4. Circular doubly linked list

Advantages of linked list over arrays:

1. Size of array is fixed where linked list is dynamic.
2. Insertion and deletion operations involving array is tedious job but easy in linked list.

1. Singly linked list

If there exits only one link field in each node of the list, the link list is called singly linked list.

<u>Creating node structure</u>

```
struct node

{

        int data;

        struct node *link;

};

struct node *first;

first = (struct node *) malloc (size of(struct node));
```

Operations on singly linked list

1. Inserting a node into the list
2. Deleting a node from list
3. Computing length of the list.
4. Search a list
5. Display contents of list

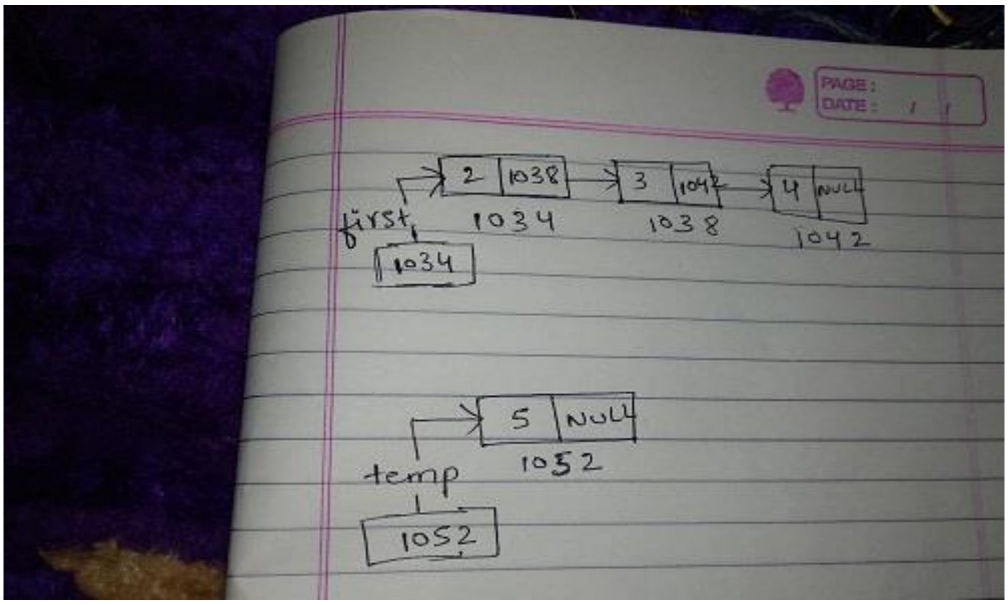
1. Inserting a node into the list

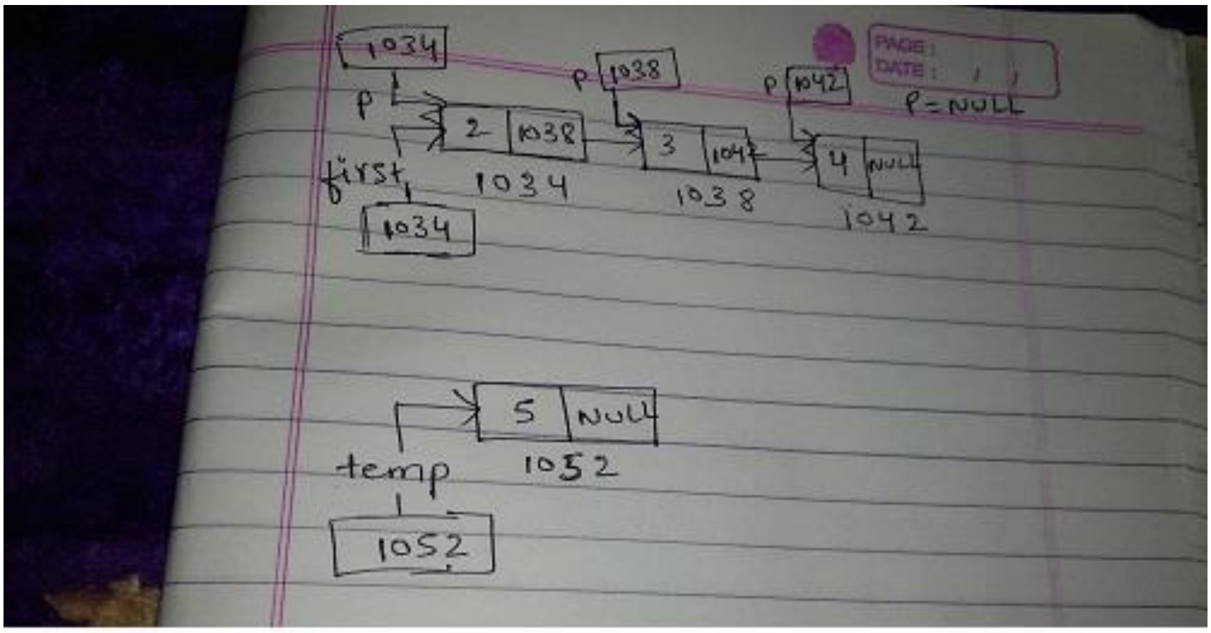
Inserting a node at the end.

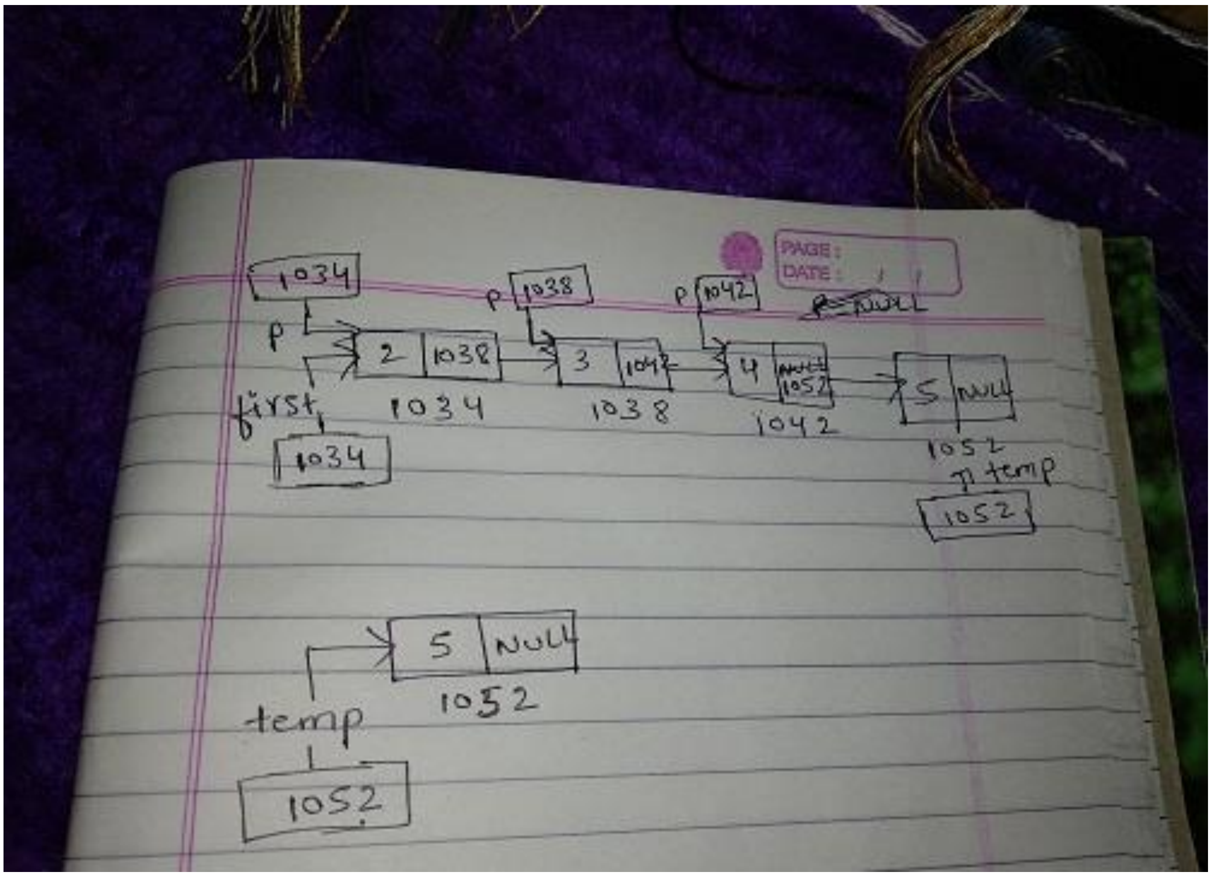
Inserting a node at the front.

Inserting node at given position.

Inserting a node at the end:





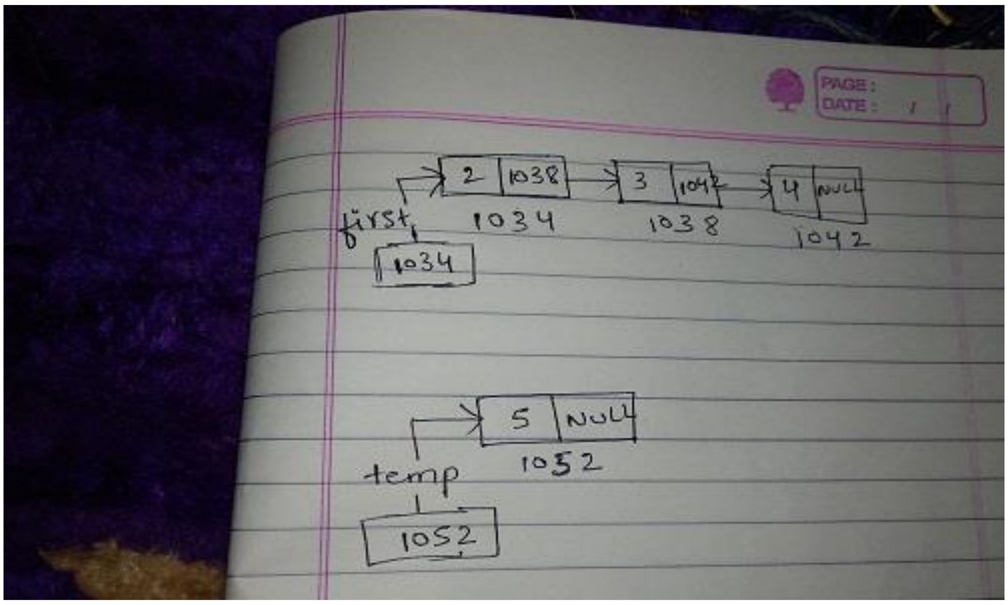


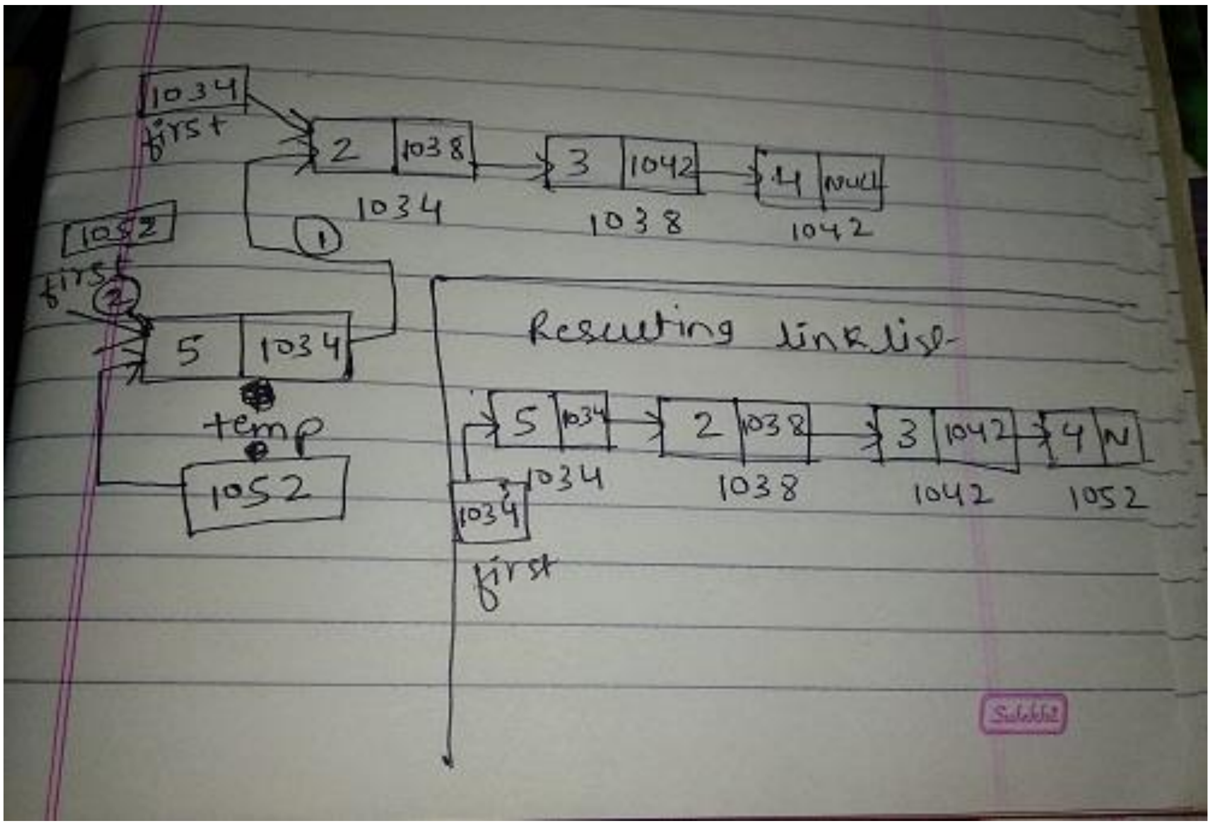
```
struct node
{
        int data;
        struct node *link;
}
struct node *first=NULL;

void append()
{
        struct node *temp;
        temp=(struct node *)malloc(size of(struct node));
        printf("enter the data element");
        scanf("%d",&temp->data);
        temp->link=NULL;
        if(first==NULL)
        {
                first=temp;
        }

        else
        {
                struct node *p;
                p=first;
                while(p->link!=NULL)
                {
                        P=p->link;
                }
                p->link=temp;
        }
}
```

Inserting a node at the front:



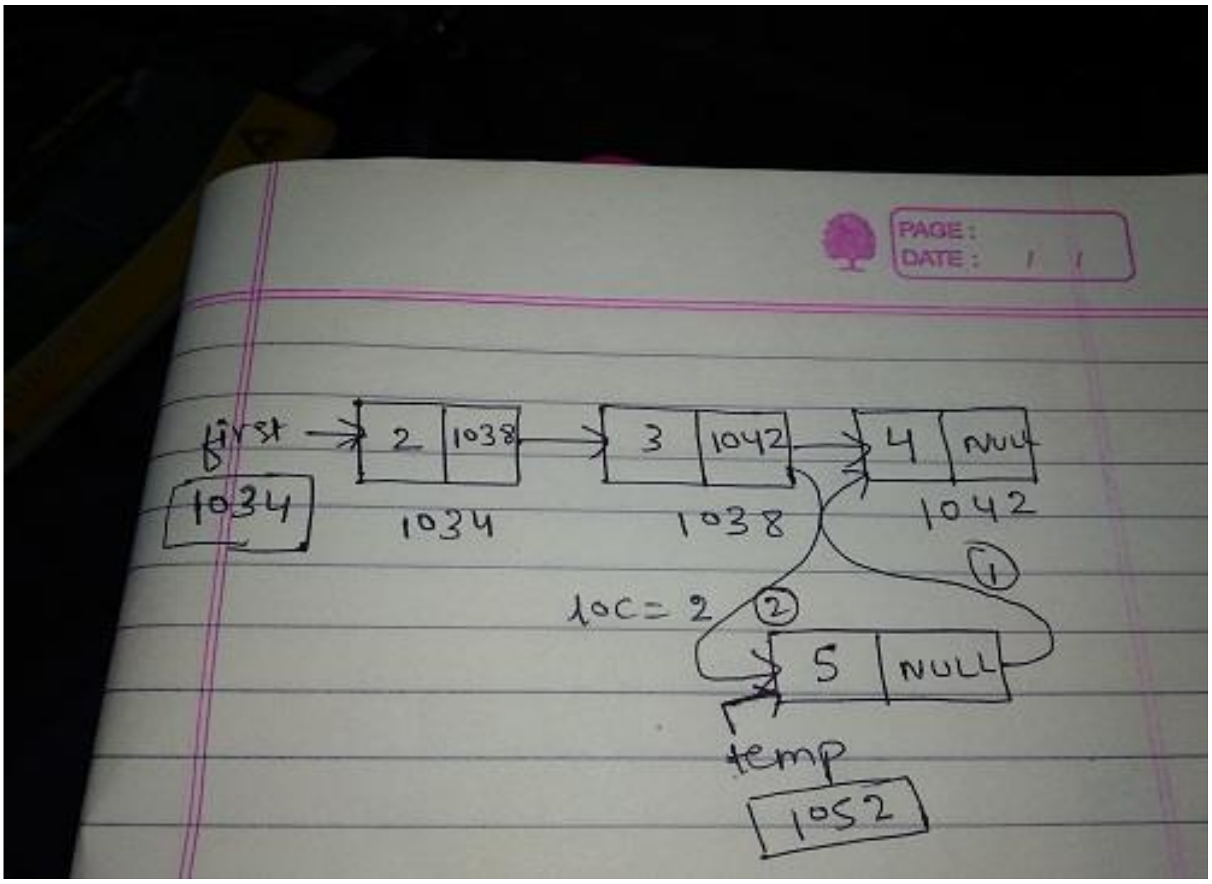


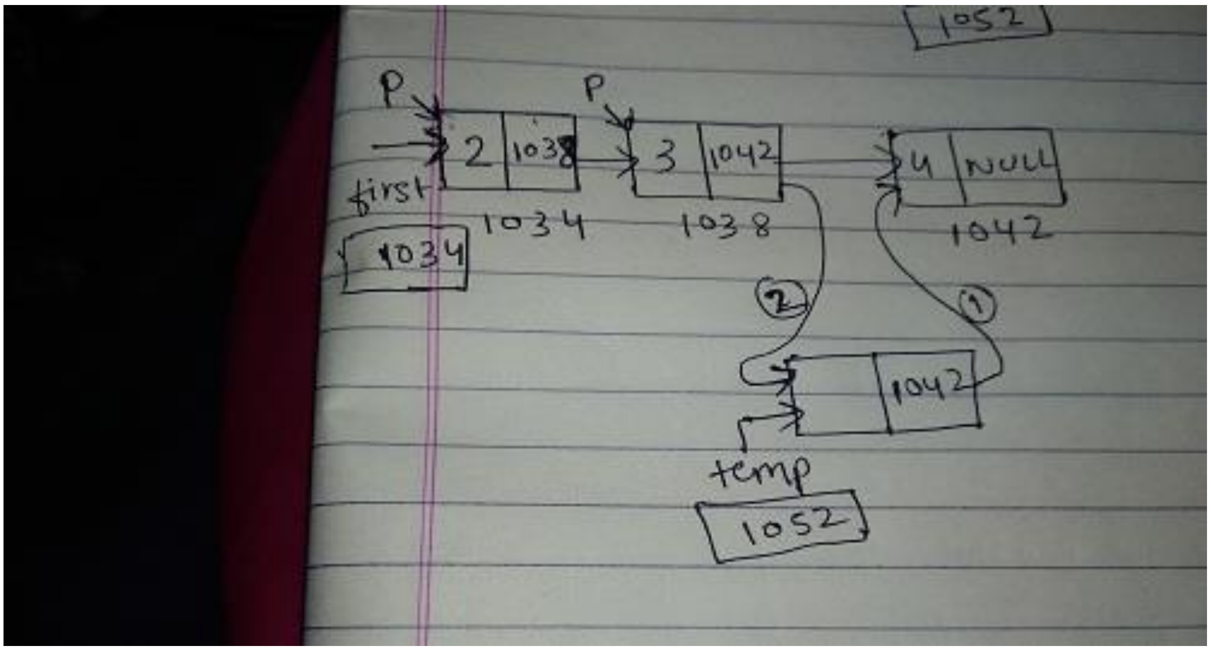
```
struct node
{
        int data;
        struct node *link;
}
struct node *first=NULL;



void insert_front()
{
        Struct node *temp;
        temp= (struct node*) malloc(sizeof(struct node))
        printf(“enter a element”);
        scanf(“%d”,&temp->data);
        temp->link=NULL;
        temp->link=first;
        first=temp;
}
```

Inserting node at given position.





Void addatposition()

{

Struct node *temp, *p;

Int loc,len,i=1;’

```
            printf(“enter location”);
            scanf(“%d”,&loc);
            len=length();
            if(loc > len)
            {
                printf(“invalid location”);
            }
            Else
            {
                P=first;
                While(i<loc)
                {
                    P=p->link;
                    i++;
                }
            temp=(struct node *)malloc(size of(struct node))
            printf(“enter a element”);
            scanf(“%d”,&temp->data);
            temp->link=NULL;
             temp->link=p->link;
            p->link=temp;
        }
    }
```