chap 3 – 4 marks

stack
stack OP
stack application
queue (linear & circular queue)
conversion (pre-fix & post-fix)

Date
Page

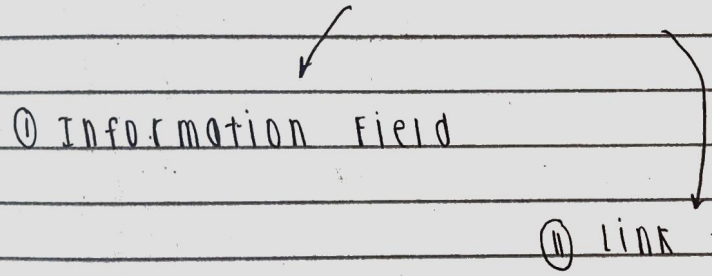
hap: 4

ast
DEC

Linked list

A linked list is a series of data items with each item containing a pointer giving a location of the next item in the list.

Each node consists of two parts



Types of linked list

① singly linked list
- linear
- circular

② doubly linked list
- linear
- circular

Basic operations in linked list

① Insertion
② Deletion
③ Traversing & display
④ searching

last ko node ko next le 1st node lai point garyo vane circular

singly linked list

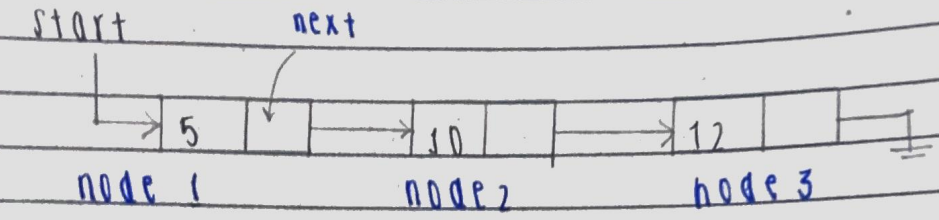


Fig: singly linear linked list

doubly linked list

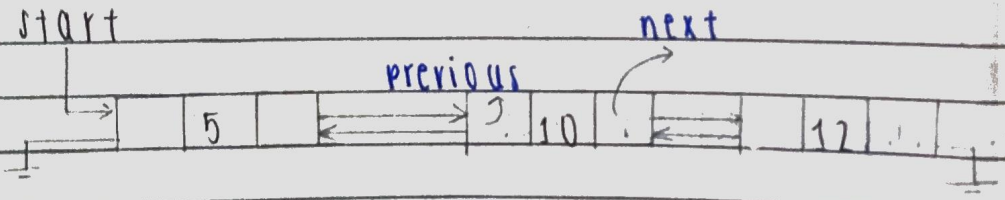


Fig: doubly linear linked list

↳. If

1st node ko previous points to last node

&

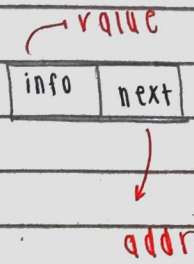
last node ko next points to 1st node

then it is circular

① singly linked

creation of linked list

```
struct list
{
    int info;
    stack list *next;
};

typedef struct list node;
node *item, *start;
```



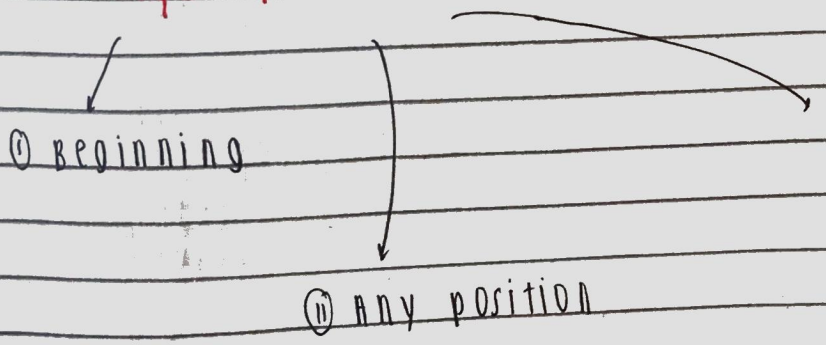
Insertion

To insert a node in a list :

→ Allocate memory to a node

→ Assign value to information field

→ Adjust ptrs to link the node at the reqd place

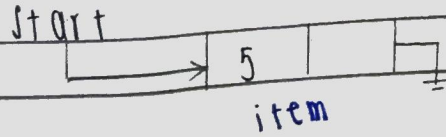


a) inserting at beginning of list

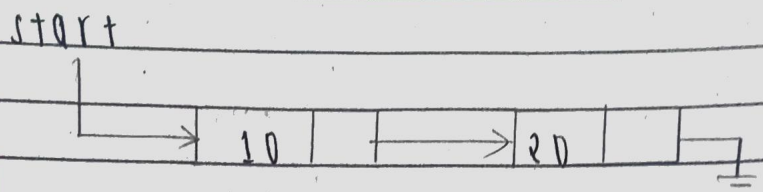
If start = NULL

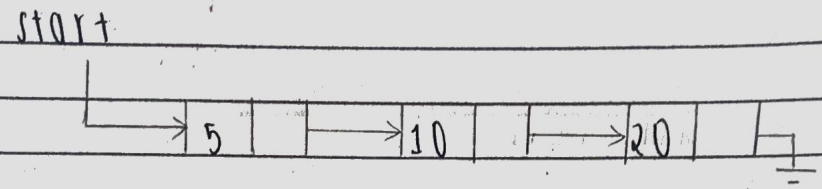
node 1st pani

last pani ho



else





- C fn

```
void f_insert ()
{
    node * item;
    item = (node *) malloc (sizeof(node));

    printf(" enter data to insert ");
    scanf ("%d", &item -> info);
}
```

item le info
(address) lai point
garirako

```
if (start == NULL)
{
    item -> next = NULL;
    start = item;
}
```

empty list

as single node ma
it is also last node

else

{

item -> next = start; [5 KO next = 10]

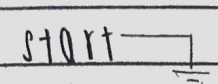
start = item; [5 at start]

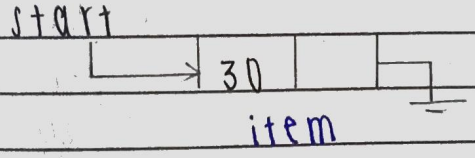
}

}

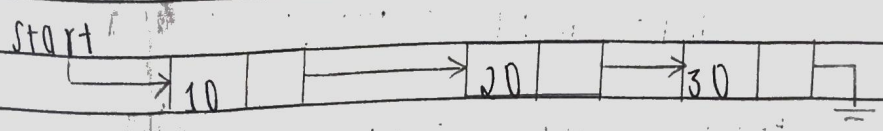
b) inserting at end of list

If



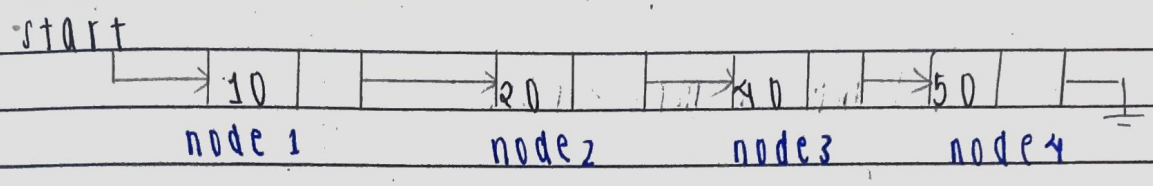


else



```
void l_insert()
{
    node *item, *temp;
    item = (node *) malloc (sizeof(node));

    printf("Enter data to insert");
    scanf("%d", &item->info);

item->next = NULL;
          ↓
        as last node ko next = NULL.

if (start == NULL)
{
    start = item;
}

else                    start bata → move gardai
{                               jane tin end
    temp = start;

    while (temp->next != NULL)   ← 10 pugna tai
    {                              2nd last
        temp = temp->next;          samma
    }                    ↳ 20 gauxa at
                                    end
 temp->next = item;
 }                  ↓
}                   then 30 insert garya
```

c) inserting node at specific position of list



Insert new node after 2

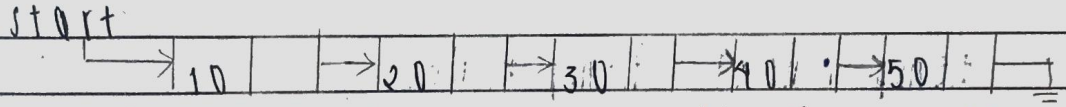


Fig: Inserting a new node (item) at specific position

C fn

```
void n_insert()
{
  int n, count = 1;
  node *item, *temp;

  item = (node *) malloc (sizeof(node));

  printf("enter the position");
  scanf("%d", &n);

  printf("enter data to insert");
  scanf("%d", &item -> info);
```

temp = start

while (count < n) ← 10 samma

{

  ++ temp = temp -> next ;

  count++;

}

↓

20 ooxo

↓

no

item -> next = temp -> next ;

↓

30 ko next

↓

40 sarxa

as temp = 20

temp->next = 40

30 ko next ma 40 sareko

temp -> next = item ;

}

↓

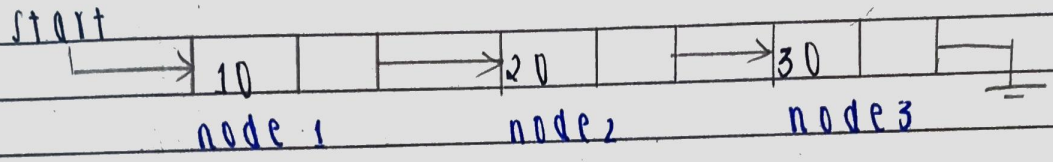
20 ko next ma 30 gako

22nd Dec

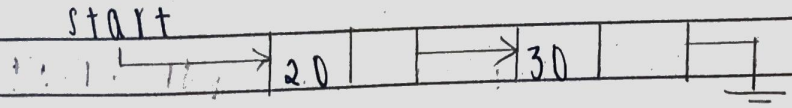
Deletion

- → Find out the node to be deleted
- → Adjust the pointer
- → Free node to be deleted

a) Deleting the first node



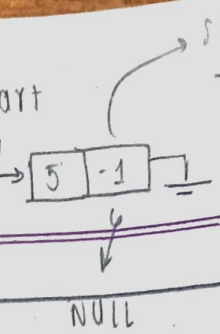
Delete node 1



C fn

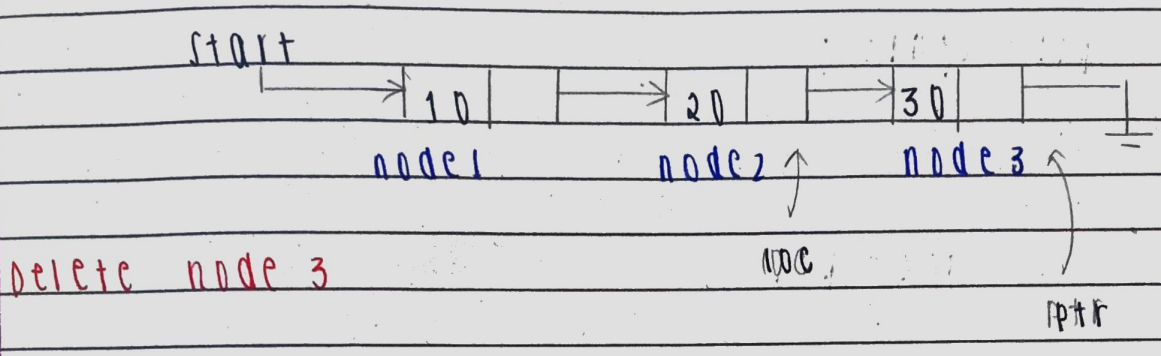
```
void f_delete()
{
    node * ptr;

    if (start == NULL)
    {
        printf(" Empty list");
        getch();
    }
```

single node wa bela

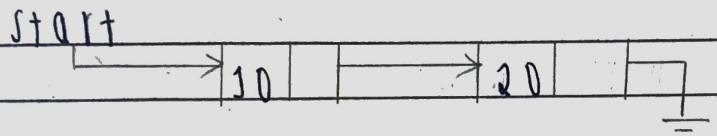



```
else if (start -> next == NULL)
{
    printf("Deleted element is %d", start -> info);
    getch();
    free (start);
        ↳ memory free gorna
    start = NULL;
}

else
{
    ptr = start;          → 20 iqi start banaso
    start = start -> next;

    printf("Deleted element is %d", ptr -> info);

    getch();
    free (ptr);
}
}
```

Deleting last node



delete node 3



C fn

```
void l_delete ()
{
   node *ptr, *loc;        // 2nd last node
                           // loi point garxa

   if (start == NULL)
   {
    printf("Empty list");
    getch();
   }

else if (start -> next == NULL)
   {
     printf(" Deleted element is %d", start->info);
     getch();

     free (start);
     start = NULL;
}
```

```
else
{
  ptr = start ;

  while ( ptr->next != NULL)
  {
        loc = ptr
```

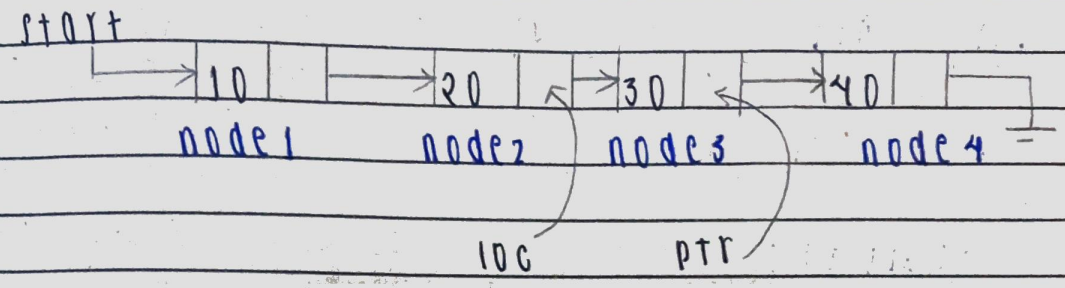
20 ko address xa

```
        ptr = ptr -> next
```

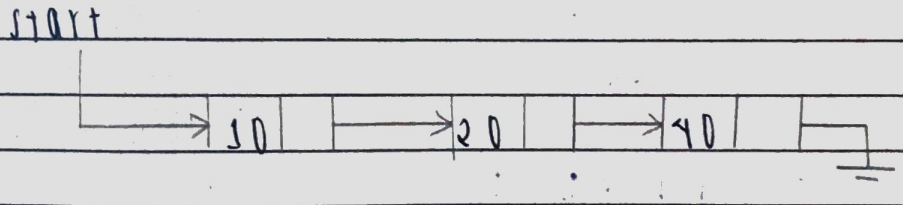
last ko aayo (30)

```
  }


loc->next = NULL;

printf ("Deleted element is %d", ptr->info);
free(ptr);
  }
}
```

c) Delete the specified portion



Delete node 3




```
c fn
  void n_delete()
{
   node *temp, *ptr;
   int n, count = 1;

if (start == NULL)
{
  printf ("Empty list");
   getch();
}

else
 {
```

```
printf("enter a position");
scanf("%d", &n);
ptr = start

while (count < n)
{
    count ++;
    temp = ptr;
    ptr = ptr -> next;
}

temp -> next = ptr -> next;

printf("deleted item is %d", ptr -> info);

free(ptr);
}
```

temp = 20

= 30

20 ko side ma 40 haleko

30 ko side ma 40 xa

28th Dec

searching

c fn

```
void search()
{
  node * temp;
  int num, count = 0;

  printf("Enter a no. to search");
  scanf("%d", &num);
  temp = start

  do
{
  count ++;

  if(temp -> info == num)
  {
  printf("This no is at %d position", count);
   temp = NULL;
   return; ~ no bhetesi program exit garne rayo
  }

else
    temp = temp -> next;    // search gareko

} while (temp != NULL);
printf("This no. doesn't exist in list");
}
```

- euta info field
- euta pointer

circular Linked list

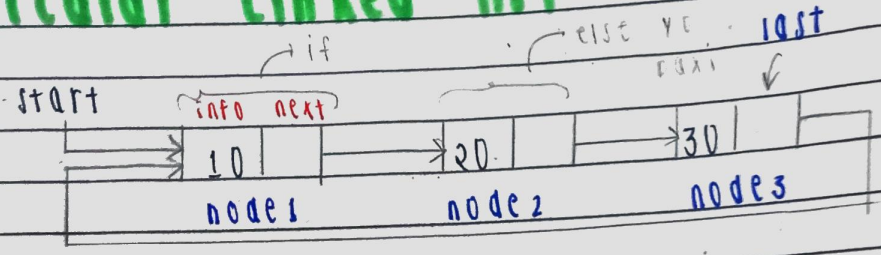


Fig: Circular linked list

```
struct list
{
  int info;
  struct list *next;
};

typedef struct list node;
node * start = NULL;
node * last = NULL;
```

- **Note:**
- last node ko next ma
  - NULL = Linear
  - 1st node = Circular

- Algorithm sodhyo vane → language, pseudo code, fig

· **Note**: Qn mei doubly linked list xa vane matra do this

& circular xa vane matra -do it

matra,

always assumed → · singly linked list
→ · linear list

· operations on singly circular linked list

1. Insertion

ⓐ beginning

ⓑ middle

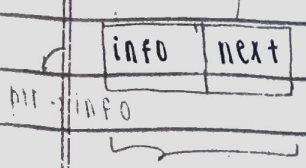
ⓒ end

a) at beginning

· Algorithm

1) Allocate memory for new node

ptr -> next ptr = (node *) malloc (sizeof (node))



2) Assign the value to information field of new node

ptr -> info = item

3) Adjust the pointer

· if => empty list
↳ start & end same node

· else => aru node xa already

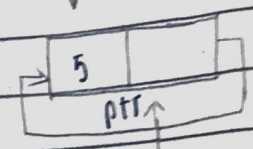
If start == NULL then

set,

ptr -> next = ptr

start = ptr

last = ptr



vesie circular banquxa

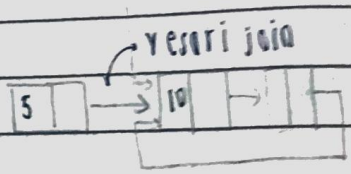
else

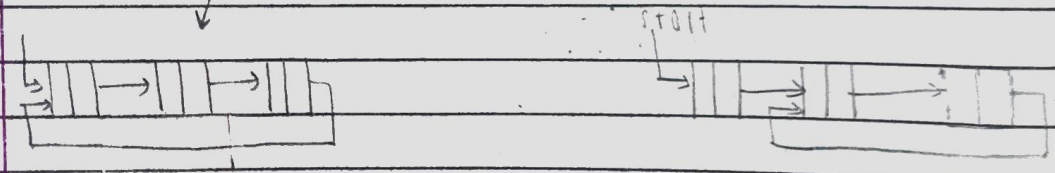
set,

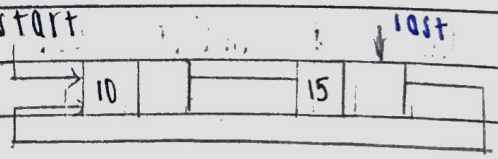
ptr -> next = start

start = ptr

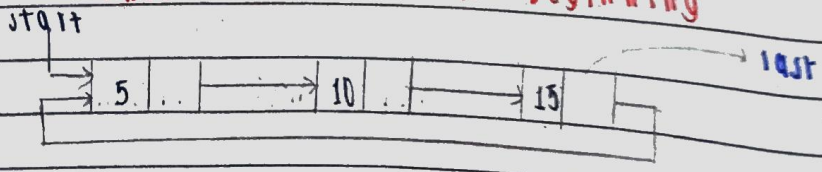
last -> next = ptr







Insert node 5 at beginning



same as beginning till 3 ko if

b) at the end

Algorithm

1) Allocate memory for new node

```
ptr = (node *) malloc (sizeof (node))
```

2) Assign value to info field of new node

```
ptr -> info = item
```

3) Adjust the ptrs

```
If start == NULL then
  set,
    ptr -> next = ptr
    start = ptr
    last = ptr
```

external ptr just holds address

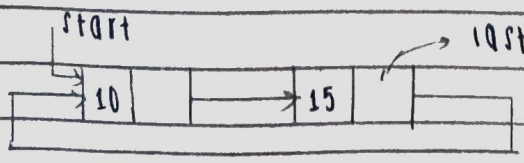
```
else
  set,
    last -> next = ptr
    last = ptr
    last -> next = start
```

15 ko next mai 20 ko address copy

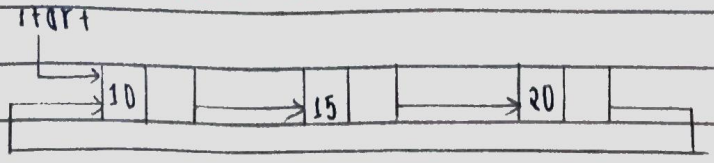
(20)

last ko value update

20 ko next mai 1st node



Insert 20 at end



Date
Page

DIY

u at specific position

2. Deletion

ⓐ beginning

ⓒ middle

ⓑ end

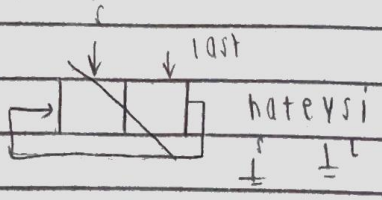
a) at beginning

· Algorithm

① If start == NULL
print "circular list in empty"

else,
follow step 2.

② If → euta matra node xa
start → next == start
print "deleted element is start -> info"
free (start)



start = NULL
last = NULL

else, → aru naxa vane
follow steps 3 to 7

③ set,
ptr = start
↓
(5)

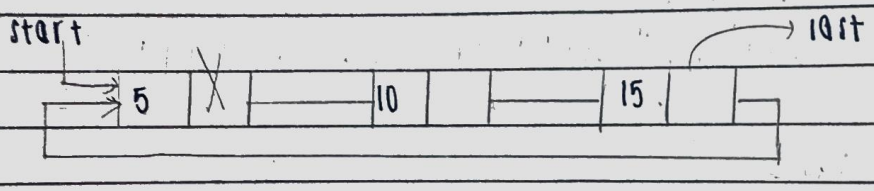
start ko update → (10 ko next)

④ start = start → next

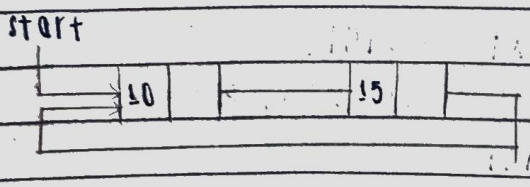
⑤ print " Deleted element is ptr → info "

⑥ last → next = start → start
↳ 15 ko last points to 10

⑦ free (ptr)
→ (5)



delete 1st node (5)



b → second last node

b) at end

Algorithm

① If start = = NULL
print " circular list is empty "

else,
follow step 2.

② If
start → next = = start
print " deleted element is start → info "
free (start)

start = NULL
last = NULL

else,
follow steps 3 to 10

③ set,
ptr = start

④ Repeat steps 5 & 6 until
ptr → next = = start
↳ last node ayesi steps

⑤ set,
temp = ptr
↳ 2nd last node

, qba last node iqi point qarxa

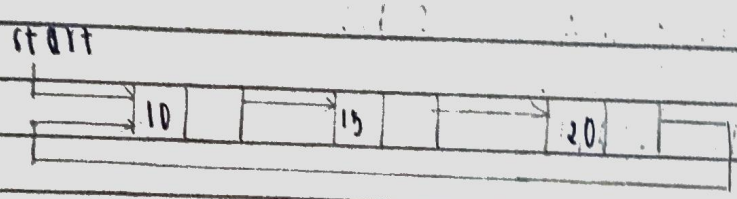
⑥ ptr = ptr → next

⑦ print " Deleted item is ptr → info"

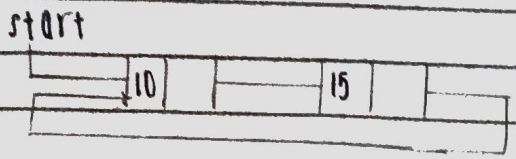
⑧ temp → next = ptr → next

⑨ last = temp

⑩ free (ptr)



Delete last node



s at n^{th} position