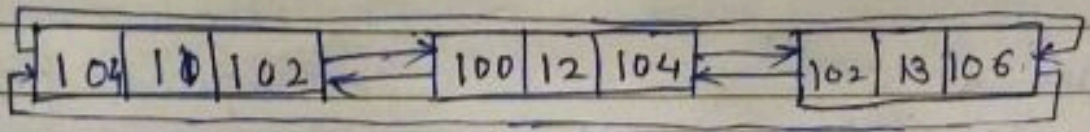
Assignment No :- 11

Q. Explain circular dll & its operations with example.

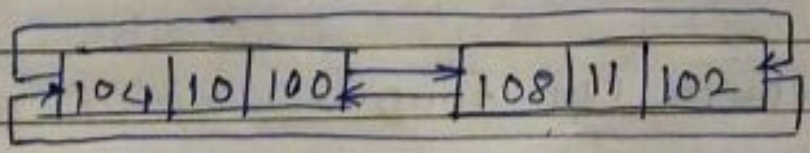
→

- Circular double linked list :-
  - Doubly circular linked list is the list in which every node contains three parts; data, previous & next.
  - Data contains actual element where as previous part contain address of previous node & next part contain address of next node.
  - The previous part of first node contains address of last node whereas the next part of last node.



- operations :-

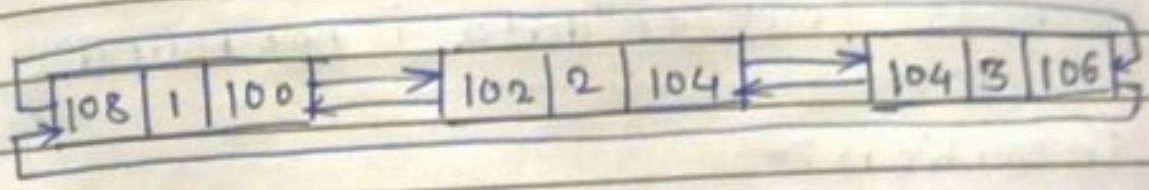
A creation :-




```
struct node
{
int data;
node* next *prev;
node* nnode;
nnode = new node;
nnode.→next = NULL;
nnode→prev = NULL;
```

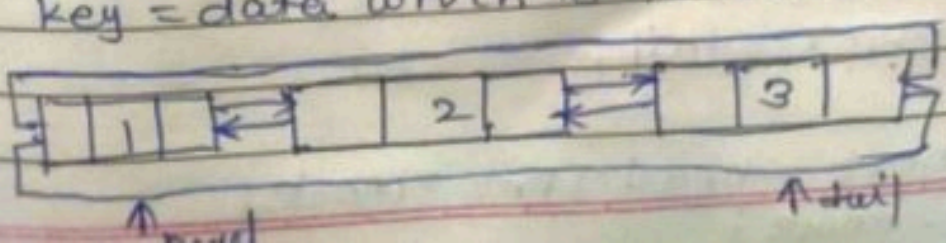
```
cout << " Enter data ";
cin >> nnode -> data;

if (head == NULL)
    head = tail = nnode;
else
{
    tail -> next = nnode;
    nnode -> prev = tail;
     tail = node;
}
```

Display linked list:-
We have to traverse linked list.




```
node *temp;
temp = head
while (temp -> data != head)
{
  cout << temp -> data;
  temp = temp -> next;
}
```

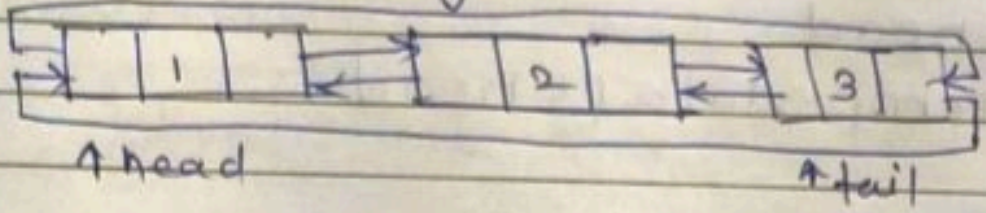
Searching:-
key = data which is to be searched.



```
node *temp;
temp = head;
flag = 0;
cout << "Enter key";
cin >> key;
while (temp != NULL)
if (temp->data == key)
{
    flag = 1
    cout << "data is found";
    break;
}
else
    temp = temp->next;
    if flag == 0
        cout << "data is not found";
```

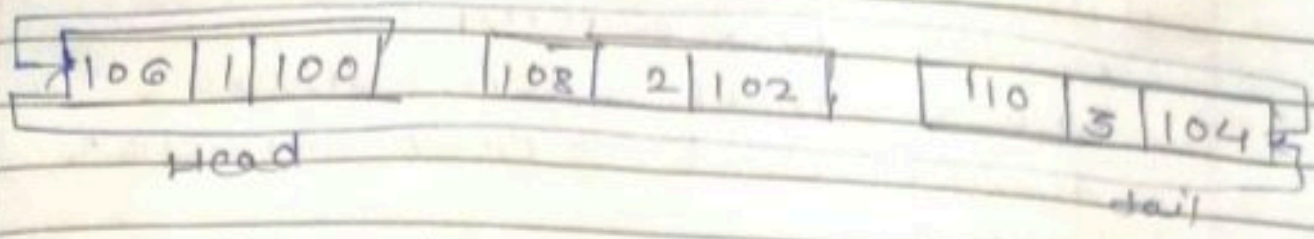
D] Insertion :-

1) At beginning :-



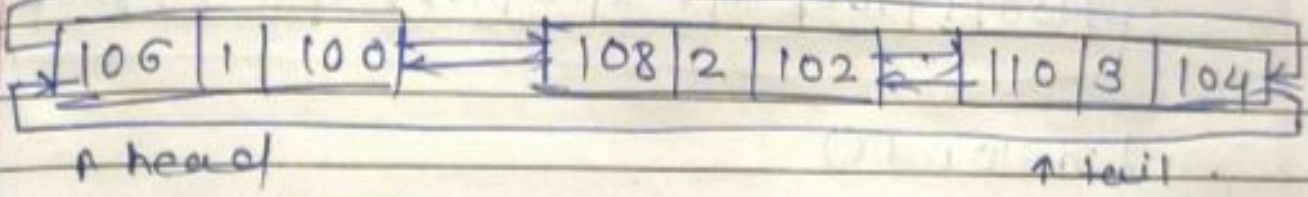

```
node *nnode;
nnode = newnode
cout << "Enter data";
cin >> nnode->data;
nnode->next = head;
nnode->prev = tail;

tail->next = head->prev = nnode;
```

2) At end:




```
node *nnode;
nnode = newnode;
Cin>> nnode ->data;
nnode ->next = head;
(head) ->prev = nnode;
nnode ->prev = tail;
tail ->next = nnode;
```

3) At Intermediate position:-

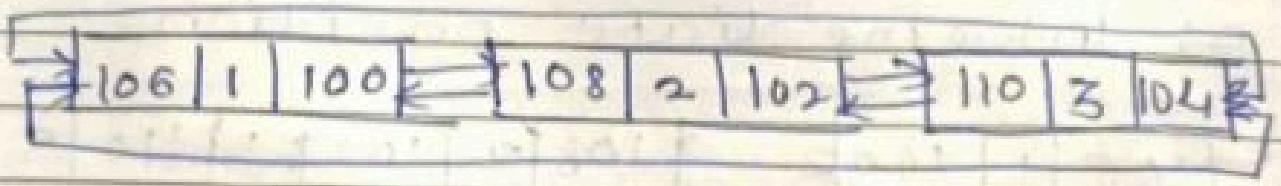



```
node *nnode = newnode;
Cout << "Enter data"; // inserting the data
Cin >> nnode ->data;

// find node value which is to be inserted
Cout << "Enter data after which u wanna
        perform insertion";
Cin >> d;
temp = head;
While (temp -> data != d)
{
    temp1 = temp -> next;
```

```
}
// Insert nnode between temp & temp1
   nnode -> prev = temp1
   nnode -> next = temp1 -> next;
   temp1 -> next -> prev = nnode;
// linking
        temp1 -> prev = nnode;
        nnode -> prev = temp1
        nnode -> prev = temp;
        temp -> next = nnode;
```

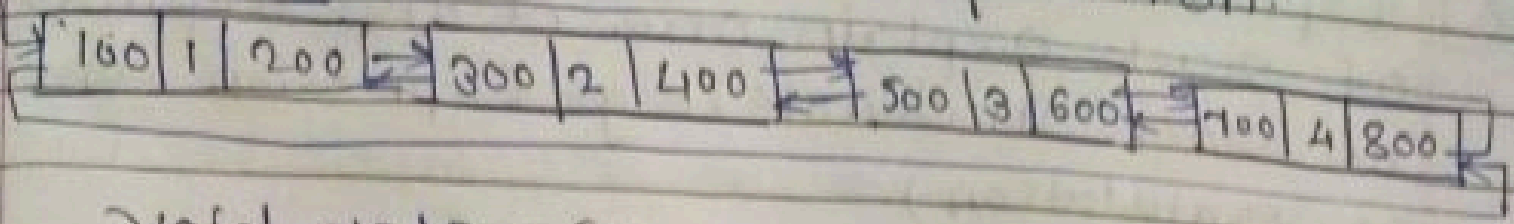
E] Deletion :-

a) deletion of head :-




```
void delend()
{ struct node * temp;
   temp = tail;
   if(head == 0)
        list is empty
   else if(head -> next == head)
        head = tail = 0;
        delet temp;
   else
      { tail = tail -> prev;
```

Delet Interme diate position:-




```
Void delpos()
{ struct node *temp;
  temp = head;
  Cout<<" Enter data ";
  Cin>>d;
  while(temp->next ->data!=d)
        temp = temp->next;
  temp->prev ->next = temp->next;
  temp->next->prev = temp->prev
  delet temp;
```