↑ ↑
curr temp

Lab program - 7

```
NODE insert~~First~~Front (NODE first){
     NODE temp;
     temp = getNode();
     int num;
     scanf("%d", &num);
     temp -> num = num;
     temp -> next = NULL;
     if(first == NULL){
          return temp;
     }
     temp -> next = first;
     first = temp;
     return first;
}

NODE deleteFront(NODE first){
     NODE curr;
     if(first == NULL){
          pf("List is empty\n");
          return NULL;
     }
     curr = first ~~}~~ -> next;
     pf("%d\n", first -> num);
```

```
        freeNode(first);
        return curr;
}

NODE sort(NODE first){
        NODE curr = first;
        NODE temp;
        if(first == NULL){
            pf("List is empty\n");
            return NULL;
        }
      while(curr != NULL){
        temp = curr->next;
        while(temp != NULL){
            if(temp->num > curr->num){
                int val = temp->num;
                temp->num = curr->num;
                curr->num = val;
            }
        }
    }
    return first;
}

NODE reverse(NODE first){
        NODE curr = NULL, temp;
        while(first != NULL){
            temp = first;
            first = first->next;
            temp->next = curr;
            curr = temp;
        }
    return temp ;

}

NODE concat(NODE first){
    NODE sec ~~{~~= NULL ;
    int chq ;
    pf("List for concatination\n");
    while(1){

        pf("Enter choice\n 1- insertfront\n 2-deletefront\n 3-display
            \n 4 - concat\n");

        scanf("%d", & chq);
        if(chq == 4){
            break;
        }
        switch(chq){
            case 1:
            sec= insertFront(~~{~~sec) ;
            break;

            case 2:
                sec = deleteFront(sec);
                break;
            case 3:
                display(sec);
                break;

        }
    }
    ~~NODE curr~~; NODE curr ;
    ~~curr = first;~~
    ~~while~~
    if(first== NULL){
        return sec;
    }
    if(sec == NULL){
        return first;
    }
    curr = first;
    while(curr -> next != NULL){
            curr = curr -> next;
    }
     curr -> next = sec;
    return first;
}
```