DSA_DAY10

Linked list :
Creating a singly linked list. Insertion, deletion, deleting from the middle

```
#include <bits/stdc++.h>

#define for0(i, n) for (int i = 0; i < (int)(n); ++i)
#define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
#define forc(i, l, r) for (int i = (int)(l); i <= (int)(r); ++i
#define forr0(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
#define forr1(i, n) for (int i = (int)(n); i >= 1; --i)

#define pb push_back
#define fi first
#define se second

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define tr(c,i) for(__typeof__((c)).begin() i = (c).begin(); i
#define present(c,x) ((c).find(x) != (c).end())
#define cpresent(c,x) (find(all(c),x) != (c).end())
#define sz(a) int((a).size())

using namespace std;

typedef vector<int> vi;
typedef vector<vi> vvi;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef long long ll;
typedef vector<ll> vll;
```

```
 typedef vector<vll> vvll;
 typedef double ld;

 struct Node {
    int val;
    Node* next;
    Node (int x){
        val = x;
        next = nullptr;
    }
 };

Node* insertAnyWhere(Node* root,int pos,int val){
    Node* newNode = new Node(val);
    if(pos == 1){
        newNode->next = root;
        return newNode;
    }
    Node* head = root;
    while(pos-- > 1) head = head->next;
    newNode->next = head->next;
    head->next = newNode;
    return root;
}

// Insetion at the end
Node*  insertEnd(Node* root, int x){
    if(root == nullptr) return new Node(x);
    Node* head = root;
    while(root->next) root= root->next;
    root->next = new Node(x);
    return head;
}

//Delete at front , any node and last node;
Node*  Delete(Node* root,int val){
```

```
    if(root->val == val) {
        Node* newHead = root->next;
        root->next = nullptr;
        delete root;
        return newHead;
    }
    Node* pre = root, *tmp = root;
    while(tmp->val != val) {
        pre = tmp;
        tmp = tmp->next;
    }
    pre->next =tmp->next;
    tmp->next = nullptr;
    delete tmp;
    return root;

}

Node* Middle(Node* root){
    Node *slow =root,*fast = root;
    while(fast->next && fast->next->next){
        slow = slow->next;
        fast = fast->next->next;
    }
    cout<<"Middle Element is "<<slow->val<<" which is goind to b
    Node* start = Delete(root,slow->val);
    return start;
}

void Display(Node* root){
    while(root){
        cout<<root->val<<" ";
        root = root->next ;
    }
    cout<<endl;
}
```

```
void solve(){
    // Create a new node
   Node* root = nullptr;
   root = insertEnd(root,1);
   root = insertEnd(root,2);
   root = insertEnd(root,3);
   root = insertEnd(root,4);
   root = insertEnd(root,5);
   cout<<"After inserting a series of nodes in single linked li
   Display(root);
   cout<<"After deleting  a single node looks like:";
   root = Delete(root,1);
   Display(root);
   cout<<"After inserting at particular position looks like:";
   root = insertAnyWhere(root,3,6);
   Display(root);
   Middle(root);
   Display(root);

}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.precision(10);
    int t=1;
    cin >> t;
    while (t--) solve();
    return 0;
}
```

Linked list :
Creating a doubly linked list. Insertion, deletion,

deleting from the middle

```
 #include <bits/stdc++.h>
#define for0(i, n) for (int i = 0; i < (int)(n); ++i)
#define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
#define forc(i, l, r) for (int i = (int)(l); i <= (int)(r); ++i
#define forr0(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
#define forr1(i, n) for (int i = (int)(n); i >= 1; --i)

#define pb push_back
#define fi first
#define se second

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define tr(c,i) for(__typeof__((c)).begin() i = (c).begin(); i
#define present(c,x) ((c).find(x) != (c).end())
#define cpresent(c,x) (find(all(c),x) != (c).end())
#define sz(a) int((a).size())

using namespace std;

typedef vector<int> vi;
typedef vector<vi> vvi;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef long long ll;
typedef vector<ll> vll;
typedef vector<vll> vvll;
typedef double ld;

struct Node {
   int val ;
   Node* left;
   Node* right;
```

```
    Node(int x){
        val = x;
        left = right = nullptr;
    }
};

Node* Delete(Node* root,int val){

    if(root->val == val){
        root->right = nullptr;
        if(root->right) root->right->left = nullptr;
        return root->right;
    }
    Node* head = root;
    while(root->val != val){
        root = root->right;
    }
    root->left->right = root->right;
    root->right->left = root->left;
    delete root;
    return head;
}
Node * insertAnyWhere(Node* root,int pos ,int val){
    Node* newNode = new Node(val);
    if(pos == 1){
        newNode->right = root;
        root->left  = newNode;
        return newNode;
    }
    Node* tmp = root;
    while(--pos > 1 ) tmp = tmp->right;
    newNode->left = tmp;
    newNode->right = tmp->right;
    newNode->right->left = newNode;
    tmp->right = newNode;
    return root;
```

```
 }
 Node*  insertAtEnd(Node* root,int val){
    if(root == nullptr) return new Node(val);
    Node* head = root;
    while(root->right) root = root->right;
    root->right = new Node(val);
    root->right->left = root;
    return head;
 }

Node* Middle(Node* root){
    Node* slow = root;
    Node* fast = root;
    while(fast->right && fast->right->right){
        slow = slow->right;
        fast = fast->right->right;
    }
    cout<<"Middle Element of DLL is "<<slow->val<<" Which is go
    Node *head = Delete(root,slow->val);
    return head;

}

 void Display(Node* root){
    Node* temp = root;
    while(temp!= nullptr){
        cout << temp->val << " ";
        temp = temp->right;
    }
    cout << endl;
 }
 void solve(){
    Node* root = nullptr;
    root = insertAtEnd(root,1);
    root = insertAtEnd(root,2);
    root = insertAtEnd(root,3);
```

```
    root = insertAtEnd(root,4);
    root = insertAtEnd(root,5);
    cout<<"After Inserting a series of Double Linked List Elemer
    Display(root);
    root = Delete(root,3);
    root = Delete(root,4);
    cout<<"After Deleting a series of Double Linked List Element
    Display(root);
    root = insertAnyWhere(root,3,6);
    cout<<"After Inserting at particular position in  a series o
    Display(root);
    root = Middle(root);
    Display(root);

}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.precision(10);
    int t=1;
    cin >> t;
    while (t--) solve();
    return 0;
}
```