**Ques-1:(**[Finding middle element in a linked list](https://practice.geeksforgeeks.org/problems/finding-middle-element-in-a-linked-list/1)**)**

#include<iostream>

using namespace std;

void insert (int);

void display (int);

void reverse (void);

struct node

{

int data;

node \*next;

};

struct node \*head = NULL;

void insert (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head;

head = newnode;

}

void display (int n)

{

int count=0;

struct node\* temp = head;

while (temp != NULL)

{

count++;

if(count==n){

cout << temp->data;

break;}

temp = temp->next;

}}

void reverse ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head = p;

}

void del (int x)

{

struct node \*temp = head;

struct node \*prev = NULL;

struct node \*nt = NULL;

while (temp != NULL)

{

if (temp->data == x)

{

if (prev == NULL)

{

head = head->next;

break;

}else

{nt = temp->next;

prev->next = nt;

break;}

} prev = temp;

temp = temp->next;}}

int main ()

{int n,t,num,x;

cin>>t;

while(t--){

cin>>n;

num=n;

head=NULL;

while(n--)

{cin>>x;

insert(x);}

reverse();

num=num/2+1;

display(num);

}

return 0;

}

**Ques-2:(**[**Reverse a linked list**](https://practice.geeksforgeeks.org/problems/reverse-a-linked-list/1)**)**

#include<iostream>

using namespace std;

void insert (int);

void display (void);

void reverse (void);

struct node

{

int data;

node \*next;

};

struct node \*head = NULL;

void insert (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head;

head = newnode;

}

void display ()

{

struct node \*temp = head;

while (temp != NULL)

{

cout << temp->data;

temp = temp->next;

}

}

void

reverse ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head = p;

}

void del (int x)

{

struct node \*temp = head;

struct node \*prev = NULL;

struct node \*nt = NULL;

while (temp != NULL)

{

if (temp->data == x)

{

if (prev == NULL)

{

head = head->next;

break;

}

else

{

nt = temp->next;

prev->next = nt;

break;

}

}

prev = temp;

temp = temp->next;

}

}

int main ()

{

int n,t;

cin>>t;

while(t--){

cin>>n;

while(n--)

{

cin>>x;

insert(x);

}

}

display ();

return 0;

}

**Ques-3:()**[**Rotate a Linked List**](https://practice.geeksforgeeks.org/problems/rotate-a-linked-list/1)

**Approach:**

1. **Create a circular linked list because circular linked list is by default has rotation property**

#include <iostream>

using namespace std;

void insert(int);

void display(int);

void reverse();

void deletee(int);

struct node

{

int data;

struct node \* next;

};

struct node\* head =NULL;

struct node\* last =NULL;

void insert(int data)

{

struct node\* new\_node=new node();

new\_node->data=data;

new\_node->next=head;

if(head==NULL)

{

last=new\_node;

}

last->next=new\_node;

head=new\_node;

}

void display(int num)

{

int count=0;

struct node\* temp=head;

struct node\* stop;

do{

count++;

if(count>num){

if(count==num+1)

stop=temp;

cout<<temp->data<<endl;

}

temp=temp->next;

}

while(temp!=stop);

}

void reverse()

{

struct node\* prev=NULL;

struct node\* curr=NULL;

struct node\* next=NULL;

curr=head;

do{

next=curr->next;

curr->next=prev;

prev=curr;

curr=next;

}while(curr!=head);

head->next=prev;

head=prev;

}

int main()

{

int n,x,num;

cin>>n;

while(n--){

cin>>x;

insert(x);

}

cin>>num;

reverse();

display(num);

return 0;

}

**Ques-4:()**[**Reverse a Linked List in groups of given size**](https://practice.geeksforgeeks.org/problems/reverse-a-linked-list-in-groups-of-given-size/1)

#include<iostream>

using namespace std;

void insert (int);

void display (int);

void display2 (int);

void reverse (void);

struct node

{

int data;

node \*next;

};

struct node \*head = NULL;

void insert (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head;

head = newnode;

}

void display (int n)

{

int count=0;

struct node\* temp = head;

while (temp != NULL)

{

count++;

if(count>n){

cout << temp->data;

}

temp = temp->next;

}}void display2 (int n)

{

int count=0;

struct node\* temp = head;

while (temp != NULL)

{

count++;

if(count<=n){

cout << temp->data;

}

temp = temp->next;

}}

void reverse ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head = p;

}

void del (int x)

{

struct node \*temp = head;

struct node \*prev = NULL;

struct node \*nt = NULL;

while (temp != NULL)

{

if (temp->data == x)

{

if (prev == NULL)

{

head = head->next;

break;

}else

{nt = temp->next;

prev->next = nt;

break;}

} prev = temp;

temp = temp->next;}}

int main ()

{int n,t,num,x,l;

cin>>t;

while(t--){

cin>>n;

l=n;

head=NULL;

while(n--)

{cin>>x;

insert(x);}

cin>>num;

display(l-num);

display2(l-num);

}

return 0;

}

**Ques-5:(**[**n’th node from end of linked list**](https://practice.geeksforgeeks.org/problems/nth-node-from-end-of-linked-list/1)**)**

#include<iostream>

using namespace std;

void insert (int);

void display (int);

void display2 (int);

void reverse (void);

struct node

{

int data;

node \*next;

};

struct node \*head = NULL;

void insert (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head;

head = newnode;

}

void display (int n)

{

int count=0;

struct node\* temp = head;

while (temp != NULL)

{

count++;

if(count==n){

cout << temp->data;

}

temp = temp->next;

}}void display2 (int n)

{

int count=0;

struct node\* temp = head;

while (temp != NULL)

{

count++;

if(count<=n){

cout << temp->data;

}

temp = temp->next;

}}

void reverse ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head = p;

}

void del (int x)

{

struct node \*temp = head;

struct node \*prev = NULL;

struct node \*nt = NULL;

while (temp != NULL)

{

if (temp->data == x)

{

if (prev == NULL)

{

head = head->next;

break;

}else

{nt = temp->next;

prev->next = nt;

break;}

} prev = temp;

temp = temp->next;}}

int main ()

{int n,t,num,x,l;

cin>>t;

while(t--){

cin>>n>>num;

l=n;

head=NULL;

while(n--)

{cin>>x;

insert(x);}

if(num>l)

cout<<-1;

else

display(num);

//display2(num);

}

return 0;

}

**Ques 6(**[**Merge two sorted linked lists**](https://practice.geeksforgeeks.org/problems/merge-two-sorted-linked-lists/1)**)**

**Approach 1:**

**By using 3rd list**

#include<iostream>

using namespace std;

void insert1 (int);

void insert2 (int);

void display ();

void reverse (void);

void reverse2 (void);

void reverse3(void);

void createlist(int);

void mergelist(struct node\*,struct node\*);

struct node

{

int data;

node \*next;

};

struct node \*head1 = NULL;

struct node \*head2 = NULL;

struct node \*head3 = NULL;

void insert1 (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head1;

head1 = newnode;

}

void insert2 (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head2;

head2 = newnode;

}

void createlist(int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head3;

head3 = newnode;

}

void display ()

{

int count=0;

struct node\* temp = head3;

while(temp!=NULL){

cout << temp->data;

temp = temp->next;

}

}

void reverse ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head1;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head1 = p;

}

void reverse2 ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head2;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head2 = p;

}

void reverse3 ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head3;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head3 = p;

}

void mergelist(struct node\* temp1, struct node\* temp2)

{

if(temp1!=NULL && temp2!=NULL){

if(temp1->data<temp2->data)

{

createlist(temp1->data);

mergelist(temp1->next,temp2);

}else

{

createlist(temp2->data);

mergelist(temp1,temp2->next);

}

}

if(temp1==NULL && temp2!=NULL)

{

createlist(temp2->data);

mergelist(NULL,temp2->next);

}

if(temp2==NULL && temp1!=NULL)

{

createlist(temp1->data);

mergelist(temp1->next,NULL);

}

}

int main ()

{int n,t,num,x,l,i;

cin>>t;

while(t--)

{cin>>n>>l;

head1=NULL;head2=NULL;head3=NULL;

for(i=0;i<n;i++)

{

cin>>x;

insert1(x);

}

reverse();

for(i=0;i<l;i++)

{ cin>>x;

insert2(x);

}

reverse2();

mergelist(head1,head2);

reverse3();

display();

}

return 0;

}

**Ques 7(**[**Pairwise swap of a linked list**](https://practice.geeksforgeeks.org/problems/pairwise-swap-elements-of-a-linked-list-by-swapping-data/1)**)**

#include<iostream>

using namespace std;

void insert (int);

void display (void);

void reverse (void);

void swappp(void);

struct node

{

int data;

node \*next;

};

struct node \*head = NULL;

void insert (int x)

{

struct node \*newnode = new node;

newnode->data = x;

newnode->next = head;

head = newnode;

}

void display ()

{

struct node \*temp = head;

while (temp != NULL)

{

cout << temp->data;

temp = temp->next;

}

}

void

reverse ()

{

struct node \*c;

struct node \*p;

struct node \*n;

c = head;

p = NULL;

while (c != NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

head = p;

}

void del (int x)

{

struct node \*temp = head;

struct node \*prev = NULL;

struct node \*nt = NULL;

while (temp != NULL)

{

if (temp->data == x)

{

if (prev == NULL)

{

head = head->next;

break;

}

else

{

nt = temp->next;

prev->next = nt;

break;

}

}

prev = temp;

temp = temp->next;

}

}

void swappp()

{int c;

struct node \*temp = head;

while(temp!=NULL){

c=temp->data;

temp->data=temp->next->data;

temp->next->data=c;

temp=temp->next->next;

}

}

int main ()

{

int t,n,x;

cin>>t;

while (t--)

{

cin>>n;

while(n--)

{

cin>>x;

insert(x);

}

}

swappp();

reverse();

display ();

return 0;

}