Shyam S Patnaik
1905770

2) link list is palindrome or not.

```
# include <stdio.h>
  Struct node {
  int data;
  Struct node* next; };
Struct node * head, * tail = NULL;
  int size = 0;
Void addNode(int data) {
Struct node * newNode = (Struct node*) malloc (Size)
  newNode -> data = data;
  new Node -> Next = Null;
  if (head == Null)
{   head = NewNode;
    tail = newNode; }
  else
  { tail -> next = newNode:
    tail = Newnode; { Size++; }

// Reverse singly linked list.
Struct node * reverse (struct node * temp)
{  Struct node * reverse Current = temp;
   Struct node * PassNode = NULL; * neat Node = NULL;
while (Current != Null)
{  nextNode = Current -> next;
   Current -> next = prev node;
   prevNode = current;
   Current = nextNode; } Return prevNode; }
```

P.T.O
⇒

```
Void isPalindrome()
{ Struct node * current = head;
    bool flag = true;
    int mid = (size%2==0) ? (size/2) : ((size+1)/2);

for (int i = 1; i < mid; i++)
    current = current->next; }
Struct node * revhead = reverselist (current->next);
while (head != NULL && revhead != NULL)
{ if (head->data != revhead->data)
{       flag = false
        break; } }
    head = head->next.
        revhead = revhead->next; }
    if (flag)
        printf ("Given singly linked list is palindrome");
    else
    printf ("Given singly linked list is not palindrome");
    }
Void display()
{ struct node * current = head;
    if (head == NULL) { printf ("List is empty");
    return; }

printf ("Nodes of list: \n");
    while (current != NULL)
{ printf ("%d", current->data);
    current = current->next; }
    printf ("\n"); }
```

```
int main()
{   addNode(1); addNode(2); addNode(3);
    addNode(2); addNode(1); display();

        isPalindrome(); return 0; }
```

Output: Nodes of ~~singly~~ list:
1 2 3 2 1
Given singly linked list is palindrome.

---

05) Reverse even positions of linked list.

```
#include <stdio.h>
#include <malloc.h>
  Struct Node
  { int data; Struct Node * next; };
Struct Node * add (Struct nodes * start, int x)
{ Stout node * new_node = (Stout node *)
                    malloc (sizeof (Stout Node));
   New_Node -> data = x;
   New_Node -> next = start;
    Start = new_node;
     return start; }
   Struct Node printing (Stout Node start)
{ Stout Node * ptr = Start;
    while (ptr != NULL)
{ printf ("%d", ptr -> data);
    ptr = ptr -> next; }
    printf ("\n"   return start; }
```

```
Struct Node reverse (struct Node start)
{ struct Node * ptr; ptr = start;
  Struct Node * prev = NULL;
  Struct Node * temp;
  while (ptr != Null)
{ temp = ptr->next; ptr -> next = prev;
  prev = ptr; ptr = temp; }
  start = prev; return start; }
  int main ()
{ struct Node * start = NULL;
Struct Node * evens = NULL; Struct Node * odds = NULL;
  int n; printf ("Enter number of elements to add:");
  Scanf ("%d", &n); for (int i = 0; i < n; i++)
{ int x; Scanf ("%d", &x);
  if (i % 2 == 1)
      evens = add (evens, x);
  else odds = add (odds, x);
  start = add (start, x); }
printf ("Given list:"); start = printing (start);
printf ("After reversing even positions of lis:");
evens = reverse (evens);
struct Node * result = NULL;
  for (int i = 0; i < n; i++)
  { int x; if (i % 2 == 0)
{ x = evens -> data;
  evens = evens -> next;
  result = add (result, x); } }

Result = reverse (result);
result = printing (result); }
```