1. **Traverse a linked list (singly linked) in a reverse order. Estimate the space complexity of the algorithm used by you.**

   Solution 1 -

```
void print_list_in_reverse(node *start)
{
node* end = NULL;
for(node* next; start != NULL; start = next)
{
next = start->next;
start->next = end;
end = start;
}


for(node* next; end != NULL; end = next)
{
next = end->next;
end->next = start;
start = end;
cout << end->data << endl;
}
}
```

   Time complexity – O (n) + O (n) = O (n)
   Space Complexity - O (1)

2. Implement a stack using a linked list. Use this stack to evaluate a polish notation (the notation and type of expression is of your choice).

Solution 2 –

```
#include<bits/stdc++.h>
using namespace std;
struct Node {
    int data;
    struct Node* next;
};

struct Node* top_of_stack;

void push(int val)
{
    struct Node* temporary;
    temporary = new Node();
    if(!temporary) {
        cout << "Memory not allotted by Operating system TRY AGAIN!";
        return;
    }
    temporary->data = val;

    temporary->next = top_of_stack;

    top_of_stack = temporary;
}

int pop()
{
    int value;
    struct Node* temporary;
    if(top_of_stack == NULL) {
        cout << "underflow occured" << endl;
        return -1;
    } else {
        temporary = top_of_stack;
        top_of_stack = top_of_stack->next;
        temporary->next = NULL;
        value = temporary->data;
        free(temporary);
    }
    return value;
}
```

```
void post_Evaluation(char expression[])
{

    int oprnd1, oprnd2;
    int res;
    for(int i = 0; i < strlen(expression); i++) {

        if(isdigit(expression[i])) {

            push(expression[i] - '0');
            continue;
        }

        else {
            if(expression[i] == '+' || expression[i] == '-' || expression[i] == '*' || expression[i] == '/') {
                oprnd2 = pop();
                oprnd1 = pop();

                if(expression[i] == '+') {
                    res = oprnd1 + oprnd2;
                    push(res);

                }

                else if(expression[i] == '-') {
                    res = oprnd1 - oprnd2;
                    push(res);
                } else if(expression[i] == '*') {
                    res = oprnd1 * oprnd2;
                    push(res);
                } else {
                    res = oprnd1 / oprnd2;
                    push(res);
                }
            }
        }

    }
    cout << top_of_stack->data << endl;
}

int main()
{

    char expression[] = "97*45*+";
    post_Evaluation(expression);

    return 0;
}
```