MINOR PROJECT

1. Evaluate a Prefix Notation Expression using Stack

```
 #include<bits/stdc++.h>
 using namespace std;
#define Operand1 op1
#define Operand2 op2
int performOperation(char op, int op1, int op2){
 switch(op){
  case '+' : return op1 + op2;
  case '-' : return op1 - op2;
  case '*' : return op1 * op2;
  case '/' : return op1 / op2;
  default : throw invalid_argument("Invalid operator");
 }
}
int evaluatePrefix(const string& expr){
 stack<int> s;
  for(int i = expr.size()-1 ; i>=0; i--){
  char ch = expr[i];

 if(isdigit(ch)){
   s.push(ch-'0');
  }

  else{
   if(s.size() < 2){
    throw invalid_argument("Insufficient Operands for operator");
   }
   int op2 = s.top();
   s.pop();
   int op1 = s.top();
   s.pop();
   s.push(performOperation(ch, op1, op2));
  }
 }
 if(s.size()!= 1){throw invalid_argument("Invalid expression");}
 return s.top();
}

int main()
{
 string InputExpression;
 getline(cin, InputExpression);

// string InputExpression = "* + 9 2 6 5";

 try{
  int result = evaluatePrefix(InputExpression);
  cout<< "Result: "<< result << endl;
 }
 catch(const invalid_argument& e){
  cerr << "Error: " << e.what() << endl;
```

```
        }

       return 0;
      }

/*

     // sample input : *+9265
        Output : Error: Invalid expression

        sample  input : -5/67
        output: -4


     */
```

2. Reverse a LinkedList in-place

```
#include <iostream>
using namespace std;

// Definition for singly-linked list.
struct ListNode {
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(nullptr) {}
};

// Function to insert a new node at the end of the linked list
void insert(ListNode*& head, int val) {
    if (!head) {
        head = new ListNode(val);
        return;
    }
    ListNode* temp = head;
    while (temp->next)
        temp = temp->next;
    temp->next = new ListNode(val);
}

// Function to reverse a linked list in place
ListNode* reverseList(ListNode* head) {
    ListNode* prev = nullptr;
    ListNode* curr = head;
    ListNode* next = nullptr;
    while (curr) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}

// Function to print the linked list
void printList(ListNode* head) {
    ListNode* temp = head;
    while (temp) {
        cout << temp->val << " ";
        temp = temp->next;
    }
    cout << std::endl;
}

int main() {
    // Taking dynamic input for the linked list
    cout << "Enter the number of elements in the linked list: ";
    int n;
    cin >> n;

    ListNode* head = nullptr;
    cout << "Enter the elements of the linked list: ";
    for (int i = 0; i < n; ++i) {
        int val;
        cin >> val;
        insert(head, val);
    }

    cout << "Original linked list: ";
    printList(head);

    // Reversing the linked list in place
    head = reverseList(head);

    cout << "Reversed linked list: ";
    printList(head);

    // Free memory
    while (head) {
        ListNode* temp = head;
        head = head->next;
        delete temp;
    }

    return 0;
}
/*
Enter the number of elements in the linked list: 5
Enter the elements of the linked list: 1 2 3 4 5



Original linked list: 1 2 3 4 5
Reversed linked list: 5 4 3 2 1

 */
```