1. (Palindrome Line)

```
using namespace std;

#include <iostream>

#include <string>


class Node

{

public:


      char data;

      Node* next;


      Node()

      {

            data = '\0';

            next = NULL;

      }

};


class Stack

{

      Node* top = new Node();


public:
```

```
Stack()
{
     top = NULL;
}

bool isEmpty()
{
     if (top == NULL)
          return true;
     else
          return false;
}

void push(char item)
{
     Node* newnode = new Node();
     newnode->data = item;
     if (isEmpty())
     {
          newnode->next = NULL;
          top = newnode;
     }
     else
     {
          newnode->next = top;
```

```
            top = newnode;
        }
}

char pop()
{
        char poped_item = '\0';
        Node* temp = new Node();
        temp = top;
        poped_item = top->data;
        top = top->next;
        return poped_item;
        delete temp;
}

void display()
{
        Node* temp = new Node();
        temp = top;
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        delete temp;
```

```
          cout << endl;
     }
};

void check_palindrome(string string)
{
     bool isPalindrome = true;
     Stack stack;
     int i = 0;

     while (string[i] != '\0')
     {
          stack.push(string[i]);
          i++;
     }

     i = 0;
     while (string[i] != '\0')
     {
          if (stack.pop() != string[i])
          {
               isPalindrome = false;
               break;
          }
          i++;
```

```
        }

        if (isPalindrome == true)
        {
                cout << endl << string << " is a palindrome :)\n";
        }
        else
        {
                cout << endl << string << " is not a palindrome :(\n";
        }
}

int main()
{
        string text;
        cout << "Enter a line of text to check palindrome:\n";
        getline(cin, text);
        check_palindrome(text);
}
```

2. (Phone Book)

```
#include<iostream>

#include<string>

using namespace std;


class List

{

private:

    int size = 0;

    struct Node

    {

        string name;

        long long number;

        Node* prev = NULL;

        Node* nxt = NULL;

        Node(string s, long long n) { name = s, number = n; }

    };

    Node* head = NULL;

    Node* tail = NULL;

public:

    void add(string s, long long n)

    {

        if (head == NULL)

        {
            head = new Node(s, n);

            tail = head;

        }

        else

        {

            Node* temp = new Node(s, n);

            tail->nxt = temp;

            temp->prev = tail;

            tail = temp;

        }

        size++;

    }


    void remove(string s, long long n)

    {

        Node* node = head;


        if (node->name == s && node->number == n)

        {

            head = node->nxt;

            node = head;

            return;

        }


        while (node != NULL)
        {
            if (node->name == s && node->number == n)
            {
                node->prev->nxt = node->nxt;
                break;
            }
            node = node->nxt;
        }
    }

    string search(string s, long long n)
    {
        int id = 1;
        Node* node = head;
        while (node != NULL)
        {
            if (node->name == s && node->number == n)
                return "Entry found at index " + to_string(id);
            node = node->nxt;
            id++;
        }
        return "NOT FOUND";
    }

    void print()
```

```
    {
        int id = 1;
        Node* node = head;
        while (node != NULL)
        {
            cout << "Entry: #" << id << "\t" << node->name <<
"\t" << node->number << "\n";
            node = node->nxt;
            id++;
        }
    }
};

void print_operations()
{
    cout << "-+-+-+-+-+-+-+-+-+-+-\n";
    cout << "operations:\n";
    cout << "add (name, number)\n";
    cout << "remove (name, number)\n";
    cout << "search (name, number)\n";
    cout << "print\n";
    cout << "-+-+-+-+-+-+-+-+-+-+-\n\n";
}

int main()
{
```

```
List phonebook;

int queries;

cout << "Enter number of queries: ";

cin >> queries;

print_operations();

while (queries--)

{

    string op, name;

    long long number;

    cin >> op;

    if (op == "add")

    {

        cin >> name >> number;

        phonebook.add(name, number);

        cout << "\n";

    }

    else if (op == "remove")

    {

        cin >> name >> number;

        phonebook.remove(name, number);

        cout << "\n";

    }

    else if (op == "search")

    {

        cin >> name >> number;
```

```
            cout << phonebook.search(name, number) << "\n\n";
        }
        else if (op == "print")
        {
            cout << "\n";
            phonebook.print();
            cout << "\n";
        }
        else
            cout << "invalid input\n";
    }

    return 0;
}
```