// John Ingram

// Test 3

// Q7

// Problem 1

class Clock {

    int hour;

    int minute;

    int second;

public:

    bool operator<(Clock otherClock) // Assume 24 hour clock because there is no am/pm time

    {

        int thisTime = (this->hour \* 3600) + (this->minute \* 60) + this->second;

        int otherTime = (otherClock.hour \* 3600) + (otherClock.minute \* 60) + otherClock.second;

        return thisTime < otherTime;

    }

};

Problem 2

|  |  |
| --- | --- |
| Items | Indices |
|  | [7] |
|  | [6] |
| 17 | [5] |
| 3 | [4] |
| 20 | [3] |
| 10 | [2] |
| 0 | [1] |
| 8 | [0] |

Top =4|V1=17|V2 =3|V3=20

Yellow items have yet to be overwritten

// Q8

// Problem 3

#define MAX 1000

class OrderedStrings {

    string items[MAX];

    int mCount; // a count of the values stored in list

public:

    OrderedStrings() { // constructor

        mCount = 0;

    }

    bool remove(string value) {

        int position;

        position = MAX + 1;

        for (size\_t i = 0; i < this->mCount; i++)

        {

            if (items[i].compare(value) == 0)

            {

                position = i;

                break;

            }

        }

        if(position == (MAX +1)) return false;

        for (size\_t i = position; i < (mCount - 1); i++)

        {

            items[i] = items[i+1];

        }

        items[MAX] = "";

        return true;

    }

};

// Problem 4

class Node { // a node for our linked list

public:

    string data;

    Node \*next;

     Node(string newvalue) { next=NULL; data = newvalue; }

};

class OrderedStrings {

    Node \*pFirst; // first node in the linked list

    int   mCount; // a count of the values stored in list

public:

    OrderedStrings() { // constructor

        pFirst = NULL;   mCount = 0;

    }

    bool remove(string value) {

        Node \*temp;

        if(pFirst->data.compare(value) == 0)

        {

            temp = pFirst->next;

            delete pFirst;

            pFirst = temp;

            return true;

        }

        temp = pFirst;

        for (Node \*i = pFirst->next; i->next != NULL ; i = i->next)

        {

            if (i->next->data.compare(value) == 0)

            {

                temp->next = i->next;

                delete i;

                return true;

            }

            temp = i;

        }

    }

};